University of Alberta

MULTI RESOLUTION ADAPTIVE OBJECT RECOGNITION SYSTEM: A STEP TOWARDS AUTONOMOUS VISION SYSTEMS

by

Ilya Levner

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computer Science

Edmonton, Alberta
Fall 2003

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisisitons et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-87807-4
Our file Notre référence
ISBN: 0-612-87807-4

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

Canada

University of Alberta

Library Release Form

Name of Author: Ilya Levner

Title of Thesis: Multi Resolution Adaptive Object Recognition System: A step towards autonomous vision systems

Degree: Master of Science

Year this Degree Granted: 2003

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

_____

Date: Sept 30/2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Multi Resolution Adaptive Object Recognition System: A step towards autonomous vision systems** submitted by **Ilya Levner** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dr. Vadim Bulitko

Dr. Russell Greiner

Dr. Glen Armstrong

Dr. David McNabb

Date approved by committee: 09/26/03

Abstract

Automation. A common word in research literature, automation reflects the desire or need to enable machines with the ability to perform mundane, repetitive, costly, and dangerous tasks currently done by human beings. This thesis, takes a *step towards automation of object recognition from digital images*. Up until the mid 1990's researchers in vision used knowledge engineering techniques to create object recognition systems. These systems performed well in the laboratory setting but proved far too brittle for real world applications due to their inability to adapt to the multitude of image variations common in the real world. Furthermore, due to large amounts of embedded domain knowledge such systems had long development cycles and could not be easily ported to other domains. Modern attempts at robust object recognition attempted to learn adaptive strategies. One such system was ADORE (Adaptive Object Recognition). The system successfully learned dynamic object recognition strategies for finding buildings in aerial images. By modelling object recognition as a Markov Decision Process, where the intermediate representations are continuous state spaces, and vision procedures are actions, ADORE was able to select the next action (i.e., vision routine) at each step so as to maximize the quality of the final image interpretation. In contrast to the majority of vision systems, ADORE did not employ a static sequence of vision routines but adaptively selected the most appropriate sequence based on image content.

As a pioneering concept, ADORE demonstrated the applicability and effectiveness of AI and machine learning techniques at automatically building vision systems. This thesis continues the research into automatic creation of object recogni-

tion systems, by investigating methods and techniques that minimize the need for human intervention while maximizing the performance and portability of a computer vision system. The novel contributions include the following. By reducing control points, the need for feature extraction and function approximation is reduced to minimal levels. In addition, feature extraction is currently performed by human domain experts. By compressing the data tokens to a size manageable by machine learning algorithms, the system can eliminate the need for hand-crafted features. Finally, to study the aforementioned approaches aimed at removing the need for human expertise in construction of vision systems, an extended version of ADORE is applied to the challenging domain of forestry.

Acknowledgements

First and foremost I would like to express my deepest appreciation to my supervisor, Dr. Vadim Bulitko. Without his patient help, insightful suggestions, unwavering dedication, and relentless drive to succeed this thesis could not have been produced. I would also like to thank my family for their love and support over these last two years.

This project would not have been possible without Dr. Bruce Draper, who provided the code from original ADORE system and a number of ideas critical to the success of the project. Special thanks go out to Dr. Greiner for his continuous support and dedication to the project. The initial motivation for automated forestry inventory system was provided by Dr. McNabb, who envisioned the iNRIIS/FIMS system. We thank him for inspiring our research. The other members of the IRCL team deserve recognition as well. Greg Lee and Lihong Li, had numerous contributions, suggestions and comments.

I would like to express my appreciation to the committee members: Dr. Russell Greiner, Dr. Glen Armstrong, Dr. Dave McNabb as well as the committee chair Dr. Rob Holte for taking time to evaluate and comment on the research conducted during my masters program.

Additional thanks go out to Dr. Terry Caelli, as well as Qiongyan Fang and Li Cheng for providing and annotating the Vegreville plantation data, respectively; Alberta Research Council and Chris Smith for providing and labeling the Fox Creek data, respectively.

Funding for this project was provided in part by the Computer Science Department at the University of Alberta; NSERC research grant; NSERC scholarship; AICML center.

Contents

1 Introduction **1**
- 1.1 Research Motivation . . . 2
- 1.2 Adaptive Object Recognition . . . 3
  - 1.2.1 State space pruning techniques . . . 5
  - 1.2.2 Feature Extraction/Automation . . . 6
- 1.3 Forestry Domain . . . 7
- 1.4 Thesis Organization . . . 8

2 Automated Forest Inventory Approaches **9**
- 2.1 A Brief Introduction to Forestry Inventory . . . 10
- 2.2 Related Research . . . 12
  - 2.2.1 Image-Based (Model-Free, Inverse) Approaches . . . 12
  - 2.2.2 Model-Based (Forward) Approaches . . . 22
- 2.3 Summary of Related Work . . . 26

3 Modern Vision Systems **27**
- 3.1 Introduction to Image Interpretation systems . . . 28
- 3.2 Detecting Volcanoes on Venus . . . 29
- 3.3 Learning Operator Parameters . . . 31
- 3.4 Learning Control Policies . . . 34
  - 3.4.1 Schema Learning System (SLS) . . . 34
  - 3.4.2 Adaptive Object Recognition (ADORE) . . . 36
- 3.5 Summary of Related Work . . . 37

4 The Framework and Novel Contributions of MR ADORE **38**
- 4.1 Introduction . . . 39
  - 4.1.1 MR ADORE Design Objectives . . . 39
- 4.2 MR ADORE Operation . . . 40
  - 4.2.1 Building an Operator Library (MDP Actions) . . . 42
  - 4.2.2 System Rewards . . . 44
  - 4.2.3 Features as Observations . . . 46
- 4.3 Extending the Framework . . . 49

4.3.1 State space pruning techniques 50
4.3.2 Learning Control Policies 52
4.3.3 Automatic Feature Extraction 55
4.4 Summary of Research Contributions 56

5 Experiments using MR ADORE **57**
5.1 Introduction 58
5.2 The Effects of State Space Pruning 58
5.3 On-line policies performance 58
5.4 Automated Feature Extraction Performance 62
5.5 Supplementary Experiments 65
5.5.1 Labeling Errors Experiment 65
5.5.2 Sun Angle Experiment 69

6 Conclusions **72**
6.1 Future Research 73
6.1.1 Hypothesis merging 73
6.1.2 Incremental Focus of Attention 74
6.2 Conclusion 76

Bibliography **77**

A Vision Operators within MR ADORE **83**
A.1 Vision Operators 84
A.1.1 Basic Operations 84
A.1.2 Image Filters 84
A.1.3 Morphological Image Processing 88
A.1.4 Cross Match Operators 91
A.1.5 Color Segmentation 92
A.1.6 Pyramid Segmentation 93
A.1.7 Histogram Equalization 93
A.1.8 Thresholding 93
A.1.9 Resize 93
A.1.10 Submit 94

B Experimental Details and General Statistics of Interest **97**
B.1 General Statistics 98
B.2 Off-line Scores for the plantation images 98
B.3 Hand-Crafted Features 101
B.4 KNN distance metrics 102

Chapter 1

Introduction

1.1 Research Motivation

Imagine the following futuristic scenario:

It is a bright sunny day above the forest. An unmanned drone effortlessly glides through the air, sending a continuous image stream of the forest scene beneath its wings to the nearest satellite in orbit. The satellite relays the received information to the nearest forest mapping center for further processing, after that detailed forest maps are sent out to various forest companies across Canada. Using the up-to-date maps, containing detailed information such as: individual tree positions, tree species, approximate tree height and the expected yield of harvestable wood, the logging manager selects the set of trees to be harvested today. The instant the daily harvest schedule is finalized, the system dispatches the nearest mechanized harvester to log the designated trees with minimal environmental damage and reduced disturbance to the eco-system.

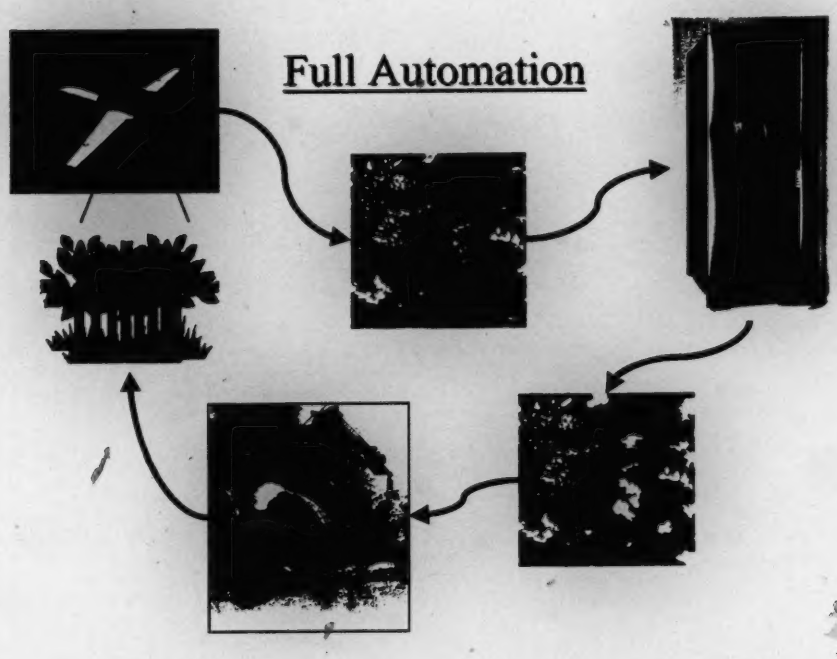


Figure 1.1: A simplified view of possible future logging practices

Although the scenario may sound like science fiction, it may not be so distant after all. Unmanned aircraft have now become common place within the military (Figure 1.1). Several Scandinavian countries currently use automated harvesters for logging purposes together with *manually* compiled forest maps. One of the major

components preventing the realization of the introductory scenario is automated image interpretation.

In fact, many other domains ranging from military and security surveillance systems to medical diagnosis of cancer regions from Magnetic Resonance Imaging (MRI) can benefit from automated vision systems. However, research within vision has been primarily focused on components or subsystems of the overall object recognition process. The comparatively few attempts at building complete systems have been primarily based on the knowledge engineering approaches common in classical Artificial Intelligence systems of the last century. Such knowledge based systems worked well in the laboratory settings but proved far too brittle in real world conditions. The tightly controlled laboratory environments enabled researchers to tailor the algorithms to the properties of their test images. However, such an *ad hoc* approach resulted in systems that could only perform within a narrow set of operating parameters atypical of real world scenarios. In addition, knowledge engineering is a time consuming process, producing systems with poor cross-domain portability that are generally expensive to maintain [Draper et al., 2000]. As a result, modern day research has shifted focus to building highly robust systems capable of being ported easily to a multitude of other domains. Such an ambitious technological milestone is yet to be accomplished. The purpose of this thesis is to attempt to bring vision systems one step closer to realization of a robust, portable system requiring little or no human intervention during its construction and operation.

1.2 Adaptive Object Recognition

One system that has recently gained attention within the vision community is ADORE (Adaptive Object Recognition) [Draper, 1996b; Draper, 1997; Draper et al., 2000; Draper et al., 2001].

As with many vision systems, ADORE identified objects in a multi-step process. Initial input data were raw images, while regions containing objects of interest represented the final output of the system; in between, the data could be represented as intensity images, probability images, edges, lines, or curves. The novelty of ADORE lies with modelling object recognition as a Markov Decision Process (MDP) [Sutton & Barto, 2000], where the intermediate representations are continuous state spaces, and the vision procedures are actions.

A Markov Decision Process can be characterized by a 4-tuple (S, A, T, R) [Boutilier et al., 1999]. Where

- $S = s_1, ..., s_n$ is a (finite) set of states the system can be in. Strictly speaking, the Markov property holds if state transition probabilities (defined below) depend only on the current state and action taken. A Markovian assumption implies that information about the current state is all that is necessary for the

agent to make an optimal action choice. Thus the agent need not concern with the history of states and actions but can compute an optimal action just from the current state information alone.

- $A = a_1, ..., a_m$ is a (finite) set of actions available to the agent. In general actions are stochastic, meaning that the execution of an action in the same state may not produce the same results over two trials.
- $T : S \times A \mapsto S$ is the state transition function defined as as $\delta(s, a, s') = P(s'|s, a)$, meaning that with some probability the agent can end up is state s' from state s by taking action a.
- $R : S \times A \mapsto \mathbb{R}$ is the reward an agent gets for being in state s and taking action a.

The goal then, is to maximize rewards over some time interval. In particular, ADORE is allowed a limited number of actions to maximize the quality of an image interpretation. Unlike single sequence systems, where a fixed/static sequence of vision procedures is used regardless of the input image, ADORE learns a *dynamic/adaptive* control policy that selects the next action (i.e., vision routine) at each step predicted to maximize the quality (i.e., reward) of the resulting image interpretation. In contrast to the knowledge based systems, which use *ad hoc* policies designed by researchers possessing vast amounts of domain knowledge, the dynamic policy used by ADORE is *machine-learned* from training examples. Each stage of the interpretation process is associated with its own data processing level (e.g., Raw Image level, Probability Map level, Segmented Image level in Figure 1.2). A sequence of operators is rewarded or penalized depending on the quality of the produced image interpretation. Thus, a standard MDP value function can be defined for each state-action pair $\langle S, A \rangle$ as the expected maximum cumulative reward the online policy can obtain by applying action A while in state S and acting optimally thereafter. In order to sample the state space all possible operator sequences of a certain length are applied off-line to a number of training image pairs, which consist of an input image and its correct output labeling. Dynamic programming is then used to compute a value function for the explored part of the state space. The sampled state values are used by machine learning methods to extrapolate them onto the entire state space. The induced value function is used within the on-line best-first control policy to select state-action pairs for execution based on the predicted value of an operator (action) applied to a given data token (state). Since raw data tokens are large and may contain irrelevant information, features must be extracted in order for machine learning methods to be successful. At least in principle, the system can be retrained to recognize objects within another domain, by simply presenting a new set of training data. In practice, however, the

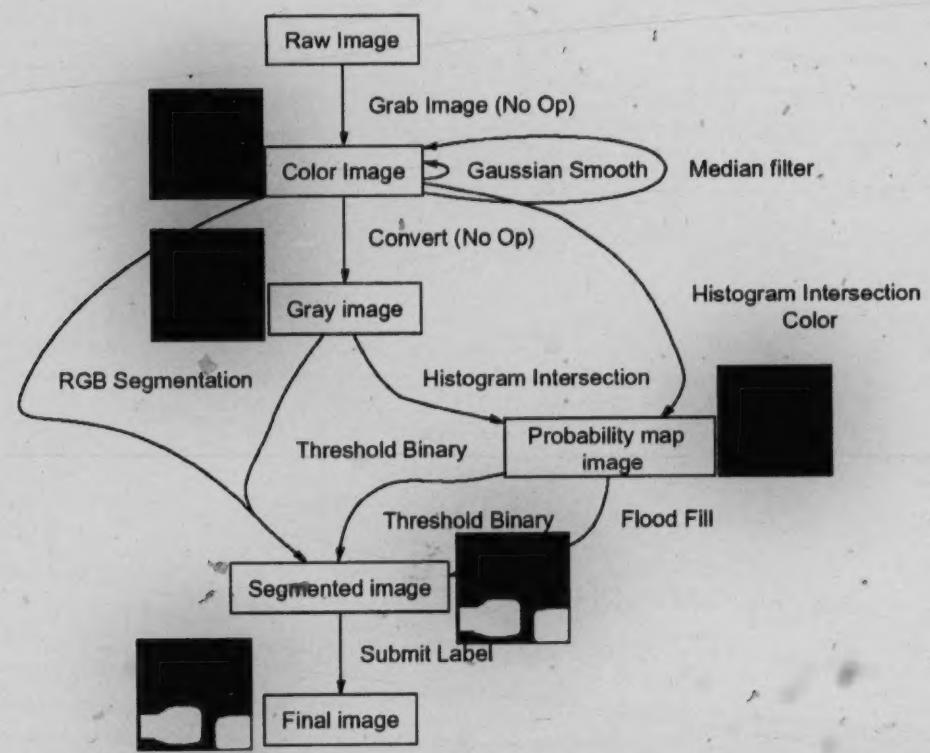


Figure 1.2: The image transformation process. Boxed nodes represent data processing levels. Labeled edges represent vision operators (actions) that transform the input data tokens to another data type. Notice several operators (e.g. Gaussian smooth) input and output the same type of data tokens.

framework of ADORE has several drawbacks and deficiencies, and as such is not fully autonomous. To further the automatic creation of vision systems several novel techniques, designed to remove the need for human intervention, are presented in this thesis. The ideas are implemented as extensions of ADORE and are evaluated in the challenging domain of forest inventory. The following sections introduce the primary novel contributions of this thesis.

1.2.1 State space pruning techniques

Both the training and operational phases of ADORE require the use of search techniques. During the off-line training stage all possible operator sequences up to a certain, user specified depth, are applied to a set of training images. However,

the system applied operators regardless of whether the sequence as a whole would produce a valid result within the allotted number of actions. In other words, the sequence length is constrained by the user, and in order to evaluate the sequence quality a terminal operator, Submit $\langle labeling \rangle$, must be applied (see Figure 1.2). Thus, executing a sequence of operators that cannot produce a valid interpretation within the permissible number of steps simply wastes resources. In response, two independent pruning techniques have been devised that eliminate the production of redundant and inadmissible data tokes.

1.2.2 Feature Extraction/Automation

One of the key factors making the framework of ADORE unique among vision systems is its dynamic decision-making abilities. In essence, the system is able to incrementally choose the appropriate sequence of operators based on the content of the input image and the intermediate data tokens. To do so, features are extracted from data tokens and used by machine learning algorithms to approximate the Q-values, which represent the optimal reward a system can achieve by applying a given vision operator (action) to a given data token (state) and acting optimally thereafter. One of the drawbacks of ADORE is that features were *hand-engineered*. That is, informative features, needed by the machine learned Q-function approximators, were designed by a domain expert. To compound the problem, each data processing level required its own set of features. For example in Figure 1.2, Color Image, Probability Map Image, Gray Image data processing layers would all require their own, possibly unique, sets of features. The constant need for hand-engineered features remains one of the last major vestiges of human intervention in ADORE. In response, the following framework extensions have been developed to reduce/remove the dependence on hand-crafted (i.e., hand-engineered) features.

Reduction of control points

Data level features and a machine-learned function approximator constitute an instance of a *control point* within the framework of ADORE. In essence the job of a control point is to enable the system to choose between different actions available to the system an a given state. For instance, the input image can be converted to grayscale, or filtered by a Gaussian kernel, or have a multitude of other actions performed on it. The job of the function approximator, using the features extracted from the input image, is to output a Q-value for a given action in order for the system to determine which action to execute. Thus the Q-values for the available actions are seen as the outputs of a single control point by the system. The more data layers and actions a system has the more control points are needed. However, one can eliminate all but one of the control points thereby sacrificing on-line ef-

ficiency (but not accuracy), while reducing the need for hand crafted features and machine-learned function approximators. In our experiments, the new system, MR ADORE, outperformed any static sequence possible with a given library of operators, just by using a single control point that requires only one set of features and only one function approximator.

Automated feature extraction

Since current machine learning techniques are unable to deal with data tokens consisting of thousands or even millions of pixels, states (i.e., data tokens) need to be abstracted by the process of extracting relevant features that compactly describe a given state. Therefore, unless a static policy is used, all control points present within the system require features. As previously mentioned, the need for hand-crafted features remains the most substantial barrier to realizing a fully automated object recognition system. Using Principle Component Analysis (PCA) [Sirovich & Kirby, 1987] raw data tokens can be compressed into a feature vector of projection coefficients. Thus, instead of requiring a domain expert to hand-engineer features, the PCA algorithm, which has been successfully used in numerous object recognition applications, can automatically extract the relevant features. By using the k-nearest neighbors, case-based learning algorithm [Duda & Hart, 1973] (along with a corpora of examples) control points can be created automatically. While the approach shows potential, it was outperformed by other control policies presented in this thesis.

1.3 Forestry Domain

For the empirical evaluation of the proposed design automation techniques, the challenging domain of *automated* forest resource inventory is used. The chosen task, extraction of individual tree parameters from aerial imagery, is motivated by the abundance of previous attempts but a lack of a robust solutions to date. Current approaches for interpretation of forest scenes from aerial images, use carefully crafted algorithms that take time and expertise to develop. More importantly, the algorithm development process exploits domain properties, such as tree separation, canopy overlap, whether the ground is flat or mountainous, or knowledge about the composition of species within the underlying forest scene. Similarly, assumptions or knowledge about the type and position of the data acquisition equipment is also integrated in the system design. Consequently, such systems work within a narrow set of operating conditions, and require substantial re-engineering in order to be applied under different settings. Such problems are not entirely new nor are they specific to the forestry domain. Knowledge-based systems within vision research have led to exactly the same problems, resulting in a shift towards more adaptive

frameworks. In this thesis MR ADORE is applied to the domain of forestry as a first stage of the interpretation process, namely supervised segmentation, and is shown to outperform any static sequence possible given a library of vision procedures.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 provides further motivation and applications for forest maps at the level of individual trees. A review of related work within the forestry domain follows. Chapter 3 presents the related research on adaptive image interpretation systems within the field of vision. The chapter also presents a thorough introduction to Adaptive Object Recognition framework. In addition to presenting the design of the Multi-Resolution Adaptive Object Recognition system, Chapter 4 describes the aforementioned framework extensions. Chapter 5 presents experimental results of running the new system on synthetic and real images of forest scenes. Finally, chapter 6 presents a discussion of future work and concluding remarks.

Chapter 2

Automated Forest Inventory Approaches

2.1 A Brief Introduction to Forestry Inventory

The first aerial image ever was acquired from an air-balloon flying over Paris in 1856, by F.Tournachin. Once airplanes took over the skies in the early part of the 20^{th} century the fields of photogrammetry[1] and remote sensing[2] started to develop. Primarily driven by silviculture[3] needs, manual interpretation of forest scenes from medium to high resolution aerial images has been used as early as 1926 [Brandtberg, 1999]. More recently the forest industry has been getting progressively interested in individual tree-level up-to-date inventory information. Crucial parameters such as: tree species, tree height, crown diameter, wood volume, and tree age class are requested/required in order to compose forestry maps and inventory databases. Such detailed information significantly aids in: managing wood logging activities (planting and cutting), ecosystem and wildlife research, as well as monitoring illegal activities. In addition knowledge about individual trees is crucial for study of forest dynamics (e.g., growth patterns, forest fire consequences and parameter correlations). Unfortunately, forest mapping at the level of individual trees is a continuous and costly undertaking. Canada alone has an estimated 344 million hectares of forests to inventory, and the goal is to re-map the boreal forests on a 10-20 year cycle [Pollock, 1996]. (Further discussion motivating the need for extraction of individual tree parameters can be found in [Gougeon & Leckie, 2003])

At such large scales, ground-based surveys and inventories are not feasible. Therefore, remote sensing techniques such as airborne or spaceborne imaging are necessary to inventory a large numbers of trees. Presently, individual tree parameters are extracted from aerial photographs by botanical and forest experts (Figure 2.1). Such manual photo-interpretation technique is a slow, labor intensive, and tedious process requiring highly skilled and sometimes highly paid domain experts. Furthermore, the procedure is error prone resulting in poor interpretation quality and subsequently poor parameter estimates. The slow and error prone manual photo-interpretation approach, requiring highly-trained domain experts, makes it impossible to scale up the process and, for example, estimate individual tree parameters for the entire province (let alone entire nation). Additionally, while interpretation textbooks and guides exist, the process is not well understood and often

[1]Photogrammetry is the science (and art) of obtaining reliable information about physical objects and the environment from photographic images and the interpretation of photographs of other remotely sensed data (e.g., Satellite Images). Interpretation of photographs is the ability to recognize and identify an object by its photographic image and the ability to describe its horizontal and vertical position from the photographs typically in a real world data and coordinate system [Thompson, 1966].

[2]Remote sensing is the science (and art) of identifying, observing, and measuring an object without coming into direct contact with it [Avery & Berlin, 1992].

[3]Silviculture is the science, art and practice of caring for forests with respect to human objectives. [Toumey & Korstian, 1947]

relies on the intuition of the interpreter built up by years of experience.

Noting the aforementioned domain constraints and the industrial push for large scale forest inventories, researchers have turned their attention to developing automated systems in order to produce forest maps from airborne images and, in the near future, LIDAR[4] and other 3D data sources. The field of *Automated interpretation of high spacial resolution digital imagery for forestry* aims to fully (or at least partially) replace the human expert by an autonomous machine able to make decisions on its own. More specifically, the goal is to measure the type (species), position, height, crown diameter, wood volume and age class for every tree in the survey area using automated approaches.

Figure 2.1: The current approach to image interpretation. A photo-interpreter using stereo-pairs of images to extract the desired information.

Such an ambitious task (automated forest mapping and inventory creation from aerial images) presents many challenges, including:

1. Massive amounts of high-resolution data, in order to recognize and measure individual tree crowns.
2. Construction and maintenance of (and providing access to) very large databases[5];
3. Geo-referencing of airborne images for validation purposes.
4. Orthorectification of aerial images, particularly given that elevation maps are often unavailable at the accuracy required.

Of particular interest are challenges created by the image content, including:

[4]LIDAR is an acronym for Light Detection And Ranging.

[5]Canada alone has on the order of 10^{11} trees.

1. Overlapping tree crowns, particularly in heavily forested regions. In addition, the shapes and sizes of tree crowns vary considerably.

2. Seasonal variations in the foliage, changing weather, and sun positions result in varying lighting and shadow conditions.

To date, no single system is able to adequately interpret all images under all the aforementioned conditions. However, under strictly controlled settings several approaches have partially succeeded in the interpretation of forest scenes.

2.2 Related Research

A number of approaches have been proposed for automatically creating forest inventories from aerial images. As with general vision applications, approaches within the forestry domain can be partitioned into the following categories. Image-based (also known as model-free or inverse) approaches use simplifying assumptions about forest images. Such approaches attempt to extract relevant image features which identify individual trees from the background foliage and ground cover. In addition features distinguishing tree species from one another are also extracted and the individual trees are then labelled with their respective class. In contrast, model-based or forward approaches model individual tree classes explicitly via 3-D models or image exemplars. These techniques aim at identifying both individual trees and their class by passing a template of the target concept over the image and maximizing a certain quality measure.

Most of the research conducted to date has focused on specific subproblems and subtasks with little research done on complete systems and their performance. The following subsections present brief overviews of subtasks and corresponding solutions within each of the two aforementioned approaches. In addition, we discuss the limited research that has been conducted on building complete systems capable of (semi) autonomous object recognition within the forestry domain.

2.2.1 Image-Based (Model-Free, Inverse) Approaches

The basis of model-free approaches is the assumption that certain known object characteristics can be used to distinguish and classify individual objects within a scene. The task of parameter extraction from aerial photographs, using the image-based approach, has been traditionally partitioned (by the forestry community) into several subtasks. Before individual tree parameters can be extracted/estimated, the individual trees must be identified within the aerial image. The task of isolating single trees is generally called "*A Tree Delineation procedure*". More recently the task has been split into two sub-processes: tree location/identificaiton followed by

tree delineation. Once individual trees have been identified, species identification and parameter extraction takes place. This stage corresponds to the classical pattern recognition problem consisting of feature extraction and classification stages. The next sections present the two commonly used approaches to Individual Tree Delineation, namely the local maximal filtering (LM) approach used to locate individual trees and the valley-following approach used to delineate the crown boundaries of the located trees from the background.

Local Maximal Filtering (LM)

The technique of local maximal filtering is based on the observation that under optimal conditions the center of a tree crown corresponds to the brightest pixel value within a localized window. That is, the local intensity maxima within a digital image correspond to the centers of tree crowns.

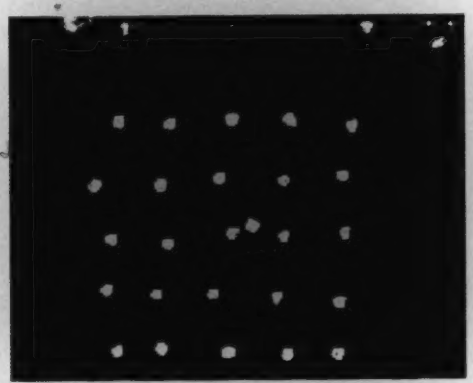


Figure 2.2: Detection of crown centers by the LM-RDA system. Yellow dots represent the approximate crown centers [Pouliot et al., 2002].

Local Maxima filtering has received a lot of attention over the last 15 years. Started in [Blazquez, 1989], LM was the method of focus in [Pinz, 1991; Rudemo, 1998; Wulder et al., 2000b; Wang et al., 2002] and numerous other papers. The research topics concentrate on finding pre and post processing procedures (such as the correct size of a Gaussian kernel used to smooth the image before LM filtering is applied [Wulder et al., 2000b; Wulder et al., 2000a; Daley et al., 2000]). Due to the fact that many other objects such as: rocks, roads, roof tops, bodies of water, grass, bushes, etc. can also produce local maxima, LM has primarily been applied to images containing only mature plantations or older natural forests that do not contain non-tree objects.

One of the earliest systems for identification of trees and their corresponding

species which uses local maximal filtering method is that of Axel Pinz. Initially described in [Pinz, 1991] and again in [Pinz, 1998b], the Austrian forest inventory system revolves around two central processing stages. First the input image is presented to a rule-based tree finding and delineation module. The purpose of the so called 'Vision Expert System' or VES (originally designed as a general purpose system for image interpretation in remote sensing) was to find the crown center and radius of every tree within the input image. Using predefined rules (i.e., hand coded sequences of image processing operators[6]) VES finds local brightness maxima that (in theory) correspond to crown peaks. After several verification steps[7] the VES outputs scene objects (in this case trees) whose image coordinates correspond with high probability to crown peaks. In addition, an estimation for the crown radius is also provided to the next stage of the object recognition process, namely species identification. Tree species recognition is done using a neural network [Haykin, 1994]. The inputs are the pixel values (red and green band only) within the mask identifying the individual tree (i.e., the red and green pixel components in a 15 by 15 pixel grid centered around the coordinates of the local brightness maxima provided by the VES stage, for a total of 450 inputs). In addition to pixel values, a locally encoded value for radius of the hypothesis mask is also a feature provided to the neural network. Such a neural net was able to achieve a testing/training performance of 93/90 %, respectively. Although no thorough experimental results are provided, the following conclusions made by the author are of utmost interest [Pinz, 1998b].

- *No single algorithm is able to solve the complete task of tree finding and species classification. A hybrid multi-level approach to scene interpretation, where several competing hypothesis are established and evaluated is required.*

- *Scale constitutes a central problem of the whole task. A fully automated system should be able to recognize trees of considerably different sizes, as well as to automatically recognize changes in image scale.*

[6]Unlike ADORE where a winner-take-all strategy is in place, VES uses all sequences and combines their output. This is possible because the sequences are hand-coded and a meaningful data fusion process can take place. Since ADORE is based on the Markovian assumption, meaning that the action depends solely on the current state, ADORE cannot merge two independently derived states into a new state.

[7]such as radial brightness distribution, (average gray values are obtained from progressively larger circles around the local maxima. The average gray values (i.e., radial brightness) are plotted against radius of the local maxima. A 'good' candidate is one where the radial brightness smoothly drops as a the distance from center increases. Several other steps are also taken to verify the spacial relations between individual local maxima co-ordinates and/or between the local maxima and other identified scene objects.

In [Pinz, 1998a], the system was used for estimating individual tree vitality and consisted of four basic steps.

1. Finding crown center coordinates.
2. Estimating crown radius
3. Tree species classification based on the two previous steps.
4. Tree vitality estimation via another neural network

Unfortunately, it seems that the Austrian government abandoned the idea of mapping the entire Austrian forest and consequently any further development of this system. Although manual interpretation is still highly in use, the author reports that the automated approach is rarely if ever used (presumably due to its poor performance and high rate of errors).

A similar technique for the near infra-red spectral band was independently developed by Gougeon and Moore in [Gougeon & Moore, 1988]. The technique, based on detecting local maxima within medium resolution images (1-3m/pixel), attempted detecting and counting mature trees. The results can be directed to a simple pixel based classifier for species recognition as well. The original technique from [Gougeon & Moore, 1988], developed for dense conifer stands with individual trees separated by shade, was later adapted to suit both high and low density plantations in [Gougeon, 1997a]. For low density stands *"a new version of the algorithm was created that looks for tree shadow at a specific distance and direction (based on sun angle) from the potential tree top" [Gougeon, 1997a].* To switch between the two algorithms a preprocessing step was introduced that looked at local gradients. For low density stands high directionality (commensurable with the direction of the sum) was exhibited. Thus a mask was created for low and high density areas used to switch between the two aforementioned algorithms.

In general, local maximal filtering seems well suited to low resolution images. When applied to images where individual trees occupy more than one pixel several local maxima can be produced for a single tree resulting in commission errors. Therefore pre and/or post processing techniques are required in order for LM to function reliably (see [Rudemo, 1998; Wulder et al., 2000b; Wulder et al., 2000a] for examples). More recently the LM-RDA system presented in [Pouliot et al., 2002] used a more refined approach to individual tree location depicted in Figure 2.3. The process started with the usual smoothing pre-processing steps followed by a fixed window LM filtering that provided the initial location of crown centers. The locations of crown centers are then adjusted by using transects (shooting rays out from the LM point) which are iteratively refined to fit the fourth order polynomial subject to a predefined error measure. The scaling procedure effectively finds

an overestimate of a tree boundary and readjusts the initial peak location to correspond to the centroid of the newly derived crown boundary. In the final step, spacial conflicts are resolved using a minimum distance filter that removes/readjusts highly overlapping crown boundaries. The delineation procedure is similar in nature to the crown detection procedure except now transect scaling is used to find the actual edge point denoting a crown boundary. The edge points are then connected using polygonal approximation to form a contour. A bank of rule based filtering procedures modifies the contours to remove improbable polygonal angles. Final outputs of the system are shown in Figures 2.2 and 2.4. This adaptive technique was shown to outperform any fixed window LM methods and requires much less parameter tuning than its predecessors.

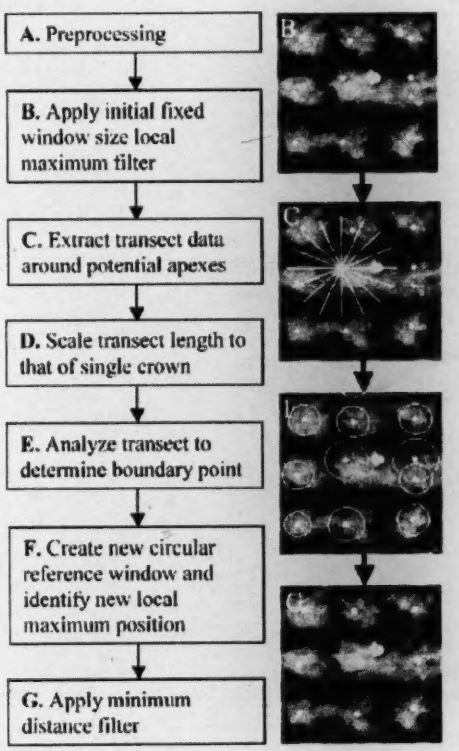

A. Preprocessing
B. Apply initial fixed window size local maximum filter
C. Extract transect data around potential apexes
D. Scale transect length to that of single crown
E. Analyze transect to determine boundary point
F. Create new circular reference window and identify new local maximum position
G. Apply minimum distance filter

Figure 2.3: Crown Location by the LM-RDA system [Pouliot et al., 2002].

Valley-following

In [Gougeon, 1993], a novel method for crown delineation was proposed. Unlike the LM method, where local image maxima are used as (possible) evidence of crown centers, Gougeon used a valley-following approach to separate trees crowns from each other and background vegetation. The valley-following algorithm looked for local minima (as opposed to local maxima) which typically exist between the individual sunlit portions of tree crowns and tried to follow these local minimums much like a river moves through the valleys between the mountain peaks. The valley-following program was run many times to maximize the connectivity of local minima. In essence the goal of the valley-following program (as the name suggests) was to follow to low intensity valleys and connect the adjacent ones together such that the individual mountains (i.e., tree crowns) were isolated from each other. This step alone, however, was not sufficient to ensure complete crown delineation (where each individual sun crown is a complete, enclosed contour). Therefore, a second stage was required to attempt the completion of the delineation process started by valley-following. The second processing phase was done by a rule-based system which tried to complete (encircle) the crown contour using a set of hierarchal rules with each level of the hierarchy designed for progressively more difficult situations which deviate from the typical clockwise valley-following path.

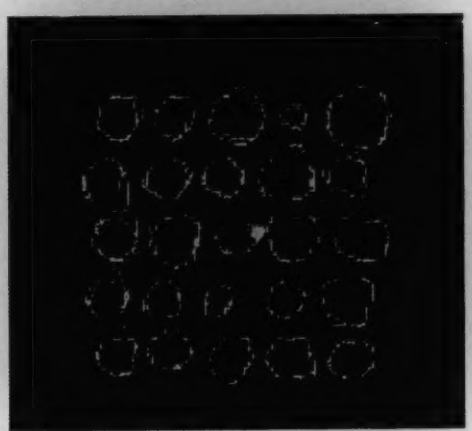


Figure 2.4: Crown Boundary Deliniation by the LM-RDA system [Pouliot et al., 2002].

Once crown delineation has been completed, individual crown diameters (or radii) could be computed and the results passed into a species recognition module [Gougeon, 1993; Gougeon, 1995a; Gougeon, 1995b; Gougeon, 1997b]. Experimental results on a *single* tree plantation image show that the outlined delin-

eation process was within 7.7% below ground-truth counts as compared to photo-interpretive results which were 18.1% below ground-truth counts[8]. However, manual screen counts (a different photo-interpretive technique) produced results of 8.5% below crown count obtained by ground based (ground-truth) counts. After examining commission[9] and omission[10] errors, the performance of the system was 81% with respect to interpretations obtained by the experienced human interpreters. However, in a later study [Gougeon, 1998], the authors found that only 58% of the automatically delineated crowns intersect (i.e., overlap) the manually delineated crowns by more than 50%. The authors readily admit that even if the error rates were brought down the approach would still fail when presented with off-nadir[11] view angles of tree canopies. In this situation the author assume that the model-based approaches (e.g., [Pollock, 1996]) would outperform the valley-following approach.

After the individual tree delineation stage has been completed an individual tree crown classification module can be applied to each delineated tree crown. Presented in [Gougeon, 1993; Gougeon, 1995a], the classification algorithm extracts spectral image features that best discriminate between tree species. In the case of [Gougeon, 1997b] average multi-spectral value for the sunlit part of the tree crown was the feature used as input into the maximum likelihood classifier[12]. The whole process consists of the following main steps:

1. Extract the near-infrared channel of the CASI[13] image.
2. Smooth the extracted data with a 3×3 Gaussian kernel.
3. Extract individual tree crowns using the valley-following algorithm followed by the rule-based crown closure procedure(s).
4. Generate/extract the spectral features identifying the species of the delineated trees. Typically the features are mean values of the red, green and blue image channels from the sunlit portion of the crown.

---

[8]This evaluation was based on raw counts of trees present within an image. Therefore, both the valley-following algorithm and photo-interpreters respectively found 7.7% and 18.1% fewer trees than those counted by the ground-based survey teams

[9]a commission error occurs when one true crown is interpreted as many crowns or an area of the image that does not contain a tree, is interpreted as containing one or more trees (i.e., false positives)

[10]omission errors occur when many true crowns are counted as one crown or a single crown is missed altogether (i.e., false negatives)

[11]nadir - having the tree trunk parallel to the view angle

[12]in [Gougeon, 1995a] the test image was 36cm/pixel MEIS-II image. Using this specific feature extracted from *hand annotated data* produced a test performance of 73.7%. On the other hand the 60cm/pixel CASI image using the same feature extracted from automatically delineated tree crowns produced a classification test accuracy of only 68.3%. The species classified were also different and therefore results are difficult to compare.

[13]Compact Airborne Spectrographic Imager. Image Resolution - 60cm/pixel

5. Train a region classifier on several hand picked examples of delineated trees. In [Gougeon & Leckie, 2001] the classifier was based on maximum likelihood decision rule[14].

In general, the performance of technique outlined above degrades as the number of tree classes increases within the image. For only 6 classes (all conifers) the system was able to classify the species of individual trees to within 12% of the ground-truth. When 11 tree species (8 conifers, 3 deciduous) are present the classification accuracy drops to 56% for conifers and 65% for deciduous species. However, " *Compared to interpreters placed in the same situation and using powerful image enhancements, results for coniferous species were inferior by 15 percentage points, but superior by 6 percentage points for the hardwoods*" [Leckie & Gougeon, 1998].

LM versus valley-following

To date one of the few applications of the two aforementioned methods (LM and valley-following) to decidious forest scene has been in [Petersen, 2001]. Experiments show that different operator parameters result in significantly different delineation results (for both LM and valley-following operators). The general conclusion made by the author is that while each algorithm is capable of providing adequate (but not optimal) delineation results, parameter settings greatly effect performance and vary from image to image.

Combining LM and Valley-following

In the preceding sections two major approaches to delineation of individual trees have been presented. First, local maxima approaches were discussed, which assume that the local maxima within the gray scale image correspond to centers of tree crowns. The second approach, valley-following, assumes that the connected local minima form a border delineating individual tree crowns from each other and the background. In [Culvenor et al., 1999] researchers combine the two aforementioned approaches into a more robust system. The TIDA (Tree identification and delineation algorithm) uses local maxima to find crown centers and valley following (along with simplified and improved post processing) to find crown contours. Tree crowns are then 'grown' starting from the local maxima outwards meeting the contours. The criteria for adding a pixel to the crown region is that the pixel value (grayscale intensity) is greater than the user specified threshold and is not a boundary pixel identified by the local minima (i.e., valley-following) algorithm.

[14]In essence, maximum likelihood is based on a matrix of probabilities or a lookup table. For a given feature value the probability of the crown being species x or species y, etc. is given. This approach was shown to outperform Artificial Neural Networks in [Gougeon, 1993]

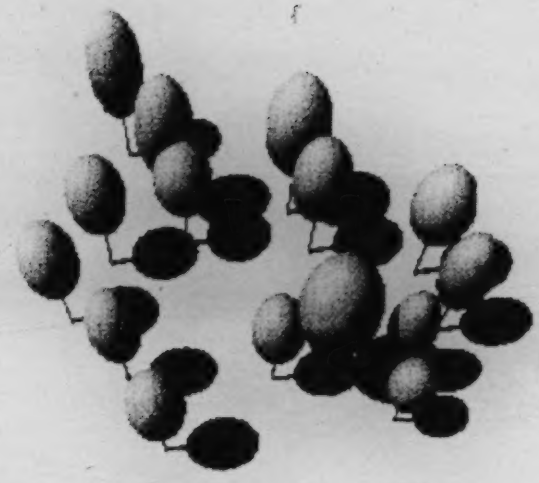


Figure 2.5: An example of a synthetic image of eucalyptus trees used in [Culvenor et al., 1999]

In addition to fusing two algorithms into a more robust tree delineation system, the experiments conducted with the new algorithm provide excellent insight into the algorithm's performance characteristics. The authors use synthetically generated data to test how well the new (hybrid) algorithm performs. Using a standard 3D tree model, where individual trees are approximated as spherical ellipsiods casting shadows on a contrasting background (see Figure 2.5), the authors respectively generate 2248 and 2285 trees having a maximal (at most, but not always) crown overlap of 35% for two series of experiments. The first experiment changed viewing angle while keeping the light source (the sun) constant, at nadir. Conversely, the second experiment kept viewing angle constant, at nadir, and varied the sun angle. In the first experiment the best result (92% of all trees were found by TIDA) occurred at a viewing angle 2 degrees of nadir. Results for the second experiment were similar with best results (of 88% of all trees found by TIDA) achieved at nadir sun position. Figure 2.6 shows complete results. These results imply that the TIDA algorithm is highly robust to changes in viewing angle, while being quite sensitive to changes in the position of the sun.

A final word on Inverse Approaches

The previous sections have discussed the inverse approaches for object recognition. The tree delineation procedures commonly used are the local-maxima filtering and valley-following both of which have similar drawbacks. Both algorithms assume

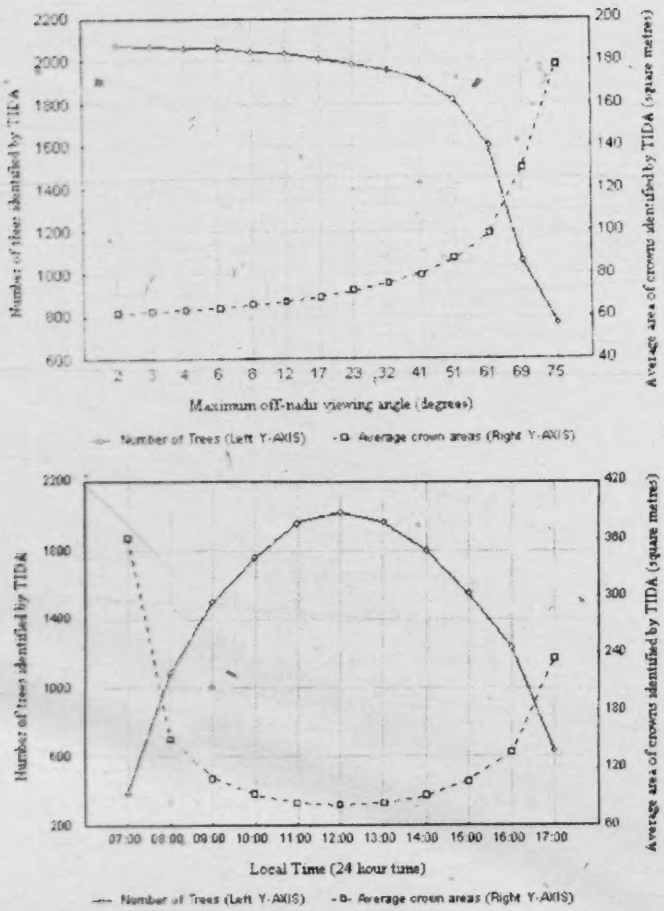

Maximum off-nadir viewing angle (degrees)
Number of Trees (Left Y-AXIS)
Average crown areas (Right Y-AXIS)
Local Time (24 hour time)
Number of Trees (Left Y-AXIS)
Average crown areas (Right Y-AXIS)

Figure 2.6: **Top:** Results for varying viewing angle and keeping the sun at nadir (represented as 12:00 in the bottom graph) angle constant. **Bottom:** Results for varying the sun angle while keeping the view angle constant (at nadir) [Culvenor et al., 1999].

a high level of contrast between the tree crown and the surrounding background. Such assumptions require specific crowding and illumination conditions in order to effectively delineate individual tree crowns. Furthermore, pre and post-processing stages needed to make these algorithms robust impose further restrictions (or the need for specific domain knowledge) on crown size distribution. More specifically, LM has a tendency to find several local maxima which in reality belong to a single tree, therefore the procedure identifies several crowns instead of one. Such a result is typically associated with high detail images filtered by a Gaussian kernel of

insufficient size during the preprocessing step. Conversely, several tree crowns (in reality) can correspond to a single local maxima resulting in a single crown being identified where several exist. This particular omission error can occur if the Gaussian kernel used to smooth the image is excessively large with respect to the size and proximity of the trees within the scene. Similar effects have been observed with the valley-following algorithm. Commission errors occur when relatively large tree crowns are separated into multiple regions (due to local minima present between the major branches of the tree) and omission errors occur within crowded tree stands.

2.2.2 Model-Based (Forward) Approaches

An alternative to inverse approaches is to use example-based image models. The underlying idea is to compare pre-specified tree crown image(s) with the image at hand. Typically such methods, e.g. [Murgu, 1996], have a collection of example tree crowns which are matched to the input image[15]. Drawbacks of such approaches include the need to collect a very large database of templates to account for differences in tree species, size, the slant of the terrain and illumination. Thus example-based approaches are most effective for images containing tree classes with little or no crown variation, such as those found on highly controlled plantations [Pollock, 1994].

To circumvent the need for a large template database, researchers have turned to model-based approaches. These approaches take advantage of an explicit model of tree crowns to match with images. One of the initial model-based systems is the STCI system developed in [Pollock, 1994; Pollock, 1996]. Unlike the example-based approaches discussed above, templates are *synthesized* from a tree crown model. The upper part of a tree crown (the so-called "sun crown") is modelled as a generalized ellipsoid of revolution and ray-tracing techniques are used to generate one or more templates [Larsen & Rudemo, 1997]. The tree crown model is created as a generalized ellipsoid of revolution defined by:

$$\frac{z^n}{a^n} + \frac{(x^2+y^2)^{\frac{n}{2}}}{b^n} = 1 \tag{2.1}$$

examples of such models are depicted in Figure 2.7. The templates are synthesized from a particular tree crown by incorporating knowledge about the sun and sensor positions as depicted in Figures 2.8 and 2.9.

The synthetic crown modelling approach has several drawbacks. The main problem is that only single channel images can be used, the selection of which, in-turn, requires a great deal of domain knowledge. Furthermore, parameters such as: sensing geometry (height, roll, pitch, flight-line azimuth, etc.); solar position

[15] A very similar system for detecting volcanoes from satellite images will be presented in chapter 3.

Figure 2.7: Examples of tree crown models produced by specified parameter settings [Pollock, 1996].

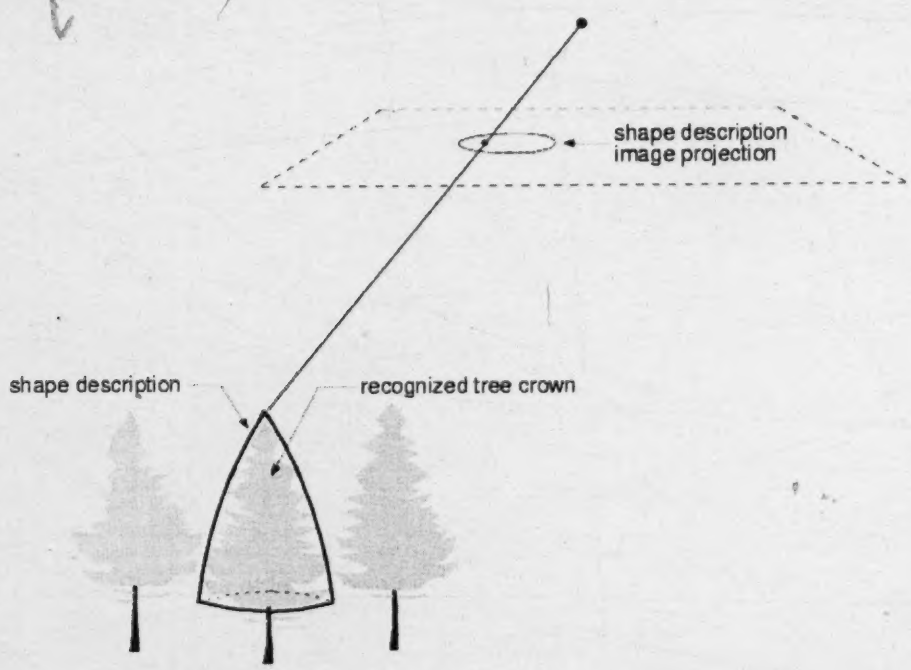


Figure 2.8: Examples of a synthetic template used to find trees [Pollock, 1996].

and light diffusion characteristics; tree crown shape, size, and diffusion characteristics; are all required in order for the system to begin processing any given image. Refer to Figure 2.9 for an example. In addition, the values of the aforementioned multitude of parameters are expected to stay constant over the image, possibly requiring additional pre-processing procedures. For example, orthorectification may be needed to create a level ground plane, which requires additional LIDAR or detailed digital elevation data in addition to the single channel image data.

Once appropriate template(s) has been generated, cross-correlation is used as a match measure. Initial hypotheses about tree locations are formed by comparing

Height of tree	22.7 m
Base of tree crown	5 m
Crown height	a=17.7 m
Crown radius	b=2.84 m
Crown shape factor	n=1.6
Foliage density (constant)	f=0.75
Foliage reflection constant	$(1 - p_{\text{bla}})K_{\text{refl}}$=0.25
Power of sun relative to sky hemisphere	1.6
Ground reflectance factors	ρ=0.3
Ground Minnaert parameter	k=1
Template radius	25 pixels≈3.75 m

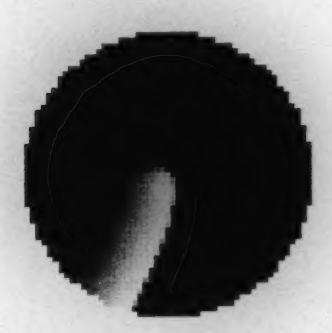


Figure 2.9: **Top:** The necessary parameters to generate a single template (**Bottom**). From [Larsen & Rudemo, 1997].

image regions with the synthesized template. Usually there will be too many positive matches between the template and an image region, requiring the use of filtering methods. Thus, a procedure similar to local-maximum filtering is used to extract local-maxima from the probability map images produced by the cross-correlation template matching procedure. Rule-based algorithms then resolve spatial conflicts removing local maximums that are spatially too close to each other. Usually that implies that the cross-correlation procedure and subsequent filtering methods identified the same tree several times.

An improved template matching algorithm presented in [Larsen, 1999] used template voting to select pixels belonging to target objects. The algorithm synthesized several templates and used each template to generate multiple initial match estimates. By thresholding individual cross-correlation scores, a per-pixel "vote" was obtained from each template match. (i.e., "*Each local maximum corresponding to a correlation coefficient of at least 0.5, ..., represents one "vote" for that position*" [Larsen, 1999].) Once again LM methods were employed to find pixels receiving a locally maximal number of votes. This particular method removed the need for selecting optimal template parameters, such as crown height, width

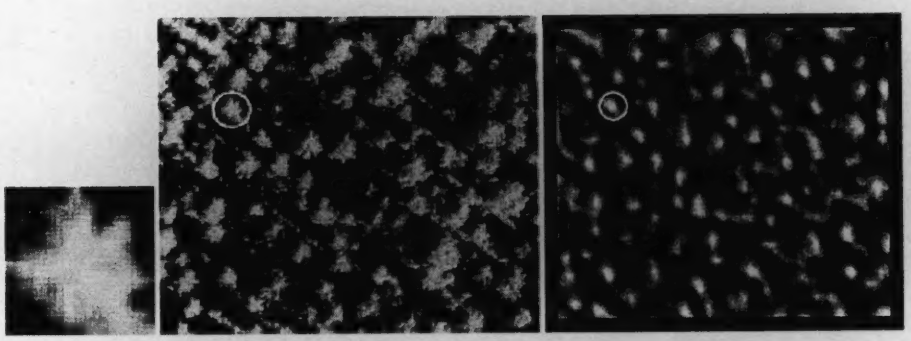


Figure 2.10: **Left-to-Right:** Template image, input image, the result of cross-correlation (between template and input image) [Korpela, 2002].

and shape factor, needed to synthesize a correct template and thereby produced a system much-more robust to real-world image variations. In a comparative study, researchers found that using multiple templates produced results equal to using hand-tuned template but without the need for human intervention. Furthermore, the algorithm eliminates the need for an *a priori* estimate of the total number of trees within a scene. Thus the multi-template approach represents a clear improvement on initial algorithms in [Pollock, 1996; Larsen, 1997].

For the most part, experiments using the synthetic tree crown have been restricted to detecting Norway spruce in a single species uniform density forests. The one exception being the original work of Pollock, who conducted experiments on mixed species plantations, achieving recognition accuracy of 56% (for all species of trees present). In contrast, experiments conducted on images containing only Norway spruce report recognition accuracy as high as 95%[16][Larsen, 1997].

[16]The experimental results presented here report the percentage of overall trees found.

2.3 Summary of Related Work

Research within the domain of forestry has branched into two directions. The model-based or forward approach to forest scene interpretation first runs a template of the target concept over the input image. The produced probability map is then subjected to several stages of filtering (e.g. Gaussian smoothing) followed by extraction of regions corresponding to image locations containing target objects. The process is concluded by resolving spatial conflicts thereby removing improbable tree positions. Alternatively, the image based or inverse approach starts out by locating individual objects and/or their boundary contours. Once the *area* occupied by individual objects has been identified each object is then classified. Each of the two approaches contains a number of algorithms, each in turn appropriate for different situations. For example, some are ideal for forest scenes containing a single tree type, while others can produce adequate results on mixed-forest stands. In general, the performance of any algorithm depends greatly on a large number of image conditions such as: sun and camera angles, tree slant, seasonal foliage variations, tree density and canopy overlap, background vegetation and terrain type, image resolution, noise, distortion, and various other parameters. To the best knowledge of the author no conclusive comparative studies have been performed to establish the relative strengths and weaknesses of algorithms presented in the previous section. Instead, research has concentrated on establishing rules and conditions necessary for a given algorithm to produce adequate results. Thus, while there is no conclusive evidence that image-based approaches perform better than model-based approaches, the latter class of algorithms does exhibit a great degree of portability. Simply changing the template can switch the target class being sought. Conversely, image-based techniques have been (and are being) developed specifically for the domain of forestry and therefore may indeed be capable of outperforming the model-based approaches in the domain of forest scene interpretation due to their single/special purpose design.

Chapter 3

Modern Vision Systems

3.1 Introduction to Image Interpretation systems

All of the approaches reviewed in the previous chapter are promising, at least in a laboratory setting, but they share some common drawbacks. For starters, they were all carefully crafted in a development process that required both time and expertise. More importantly, this development process exploited domain properties, such as whether the trees are separated or have overlapping canopies, whether the ground is flat or mountainous, or whether the forest has one type of tree or many tree species. Similarly, assumptions about the properties and position of the sensor are also integrated in the system design. As a result, these systems work within a narrow set of operating conditions, and cannot be applied under different settings without re-engineering. This is not a new problem, or a problem that is unique to the forestry domain. The same problem has plagued knowledge-based approaches to computer vision in general, see [Draper et al., 1996], and has led to new lines of research for automatically training object recognition systems. Much like the researchers within the forestry domain, researchers in vision

"have been building systems with special-purpose recognition strategies for twenty years. In the late 70s and early 80s, researchers built AI-style production and blackboard systems to select and sequence vision procedures for specific tasks. Nagao & Matsuyamas production system for aerial image interpretation [Nagao & Matsuyama, 1980] was one of the first, and lead to several long-term development efforts, including SPAM [McKeown et al., 1985], VISIONS/SCHEMA [Draper et al., 1989], SIGMA [Hwang et al., 1986], PSEIKI [Andress & Kak, 1988] and OCAPI [Clement & Thonnat, 1993]. More recently, other researchers [Chien et al., 1995; Jiang & Bunke, 1994; Lansky et al., 1995] have applied AI-style planning technology to infer control decisions from databases describing the task and the available procedures. Unfortunately, knowledge-based systems were often ad hoc. *Researchers formulated rules for selecting procedures based on their intuition, and refined these rules through trial and error. (See [Draper & Hanson, 1992] for a description of the knowledge engineering process in object recognition.) As a result, there is no reason to believe that the control policies emerging from these heuristics were optimal or even good, nor was there any way to directly compare systems or evaluate their control policies."* [Draper et al., 2000]

Based on the notion of "goal-directed vision" [Draper et al., 1996], a promising approach for autonomous system creation lies with treating computer vision as a control problem over a space of image processing operators. Initial systems, such as the Schema System [Draper et al., 1989], had control policies consisting of *ad hoc*, hand-engineered rules. While presenting a systematic way of designing image interpretation systems, the approach still required a large degree of human intervention. In the 1990's the second generation of control policy-based-image interpretation systems came into existence. More than a systematic design method-

ology, such systems used theoretically well-founded machine learning frameworks for automatic acquisition of control strategies over a space of image processing operators. To put the control of object recognition on a stronger theoretical foundation, researchers have tried using Bayes nets (e.g., TEA1 [Rimey & Brown, 1994] and SUCCESSOR [Mann & Binford, 1996]). Unfortunately, the design of Bayes nets can itself become an *ad hoc* knowledge engineering process. Other researchers tried to eliminate the knowledge acquisition bottleneck by machine-learning control policies from expert annotated examples. For instance, in [Au & Roberts, 1996] researchers used genetic algorithms to learn target recognition strategies, while reinforcement learning has been used in [Draper, 1996a] to learn control strategies or in [Peng & Bhanu, 1998] to find parameters for vision procedures. Maloof et al. train classifiers to accept or reject data instances between steps of a static sequence of procedures [Maloof et al., 1997].

This chapter presents major research on singe and multi-sequence systems. By reviewing some of the well known systems typical problems found in systems using non-adaptive image interpretation strategies are illuminated. To contrast, we review several multi-stage adaptive approaches to object recognition in order to demonstrate their benefits and costs.

3.2 Detecting Volcanoes on Venus

The paper by Michael Burl et al., titled "Learning to Recognize Volcanoes on Venus." [Burl et al., 1998] is perhaps one of the most notable works to date and is reviewed to introduce machine learning approaches for object recognition / image interpretation.

The grayscale SAR[1] images produced by the Magellan satellite orbiting Venus, are estimated to contain over 1 million small sized volcanoes. Manual delineation therefore, is expected to take several decades. To automate the process, the paper described a 3 stage system for detecting volcanoes on the Venusian surface. First, a focus of attention (FOA) algorithm was used to extract "interesting" image regions for subsequent stages of processing. Features were then extracted from the selected image regions (stage 2) and were passed to a classifier (stage 3). To extract relevant regions for later processing stages, FOA used the cross-correlation scores from running a template of the target concept over the image. The template was created by averaging the normalized examples of the target concept (provided by domain experts). The regions of interest (ROI) were passed on to the later, more computationally intensive, stages if the cross-correlation score was above the empirically derived threshold. Once FOA selected an ROI, feature extraction took place. The features were produced by projecting the ROI onto a PCA eigenbasis created

[1] Synthetic Aperture Radar

at training time from the aforementioned pool of training examples. The projection coefficients, serving as features, were then presented to a Boolean classifier. The machine learned classifier, previously trained on PCA projection coefficients of "volcano"/"non-volcano" examples, labeled the ROI. The off-line and on-line components are depicted in Figures 3.1 and 3.2, respectively.

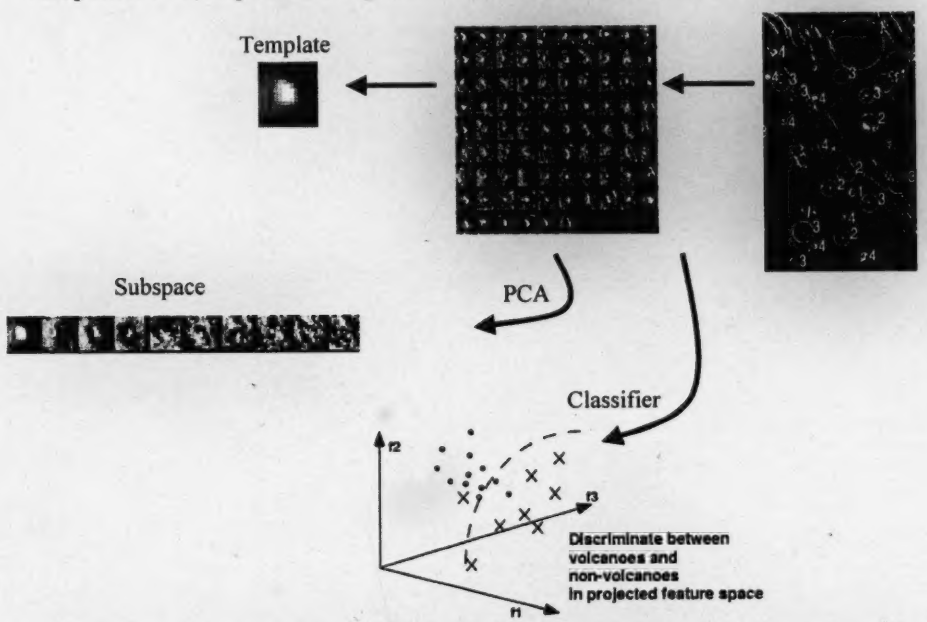


Figure 3.1: Off-line training phase of the volcano detection system [Burl et al., 1998]. First, examples of volcano objects are extracted from expert annotated images. The training images are then used to: (a) create a template of the average volcano object, (b) generate a subspace manifold via principle component analysis effectively creating a feature extractor, (c) train a classifier using features of volcano and non-volcano examples.

This method of classification achieved mixed results. On one hand, the goal of 80% correct classification was achieved when training and testing data were from the same region of the planet. On the other hand, generalization abilities were found to be limited, due to poor performance (well bellow the 80% target performance) when training and testing data were from different regions of the Venusian surface. The outlined approach closely resembles that of [Murgu, 1996], briefly mentioned in the preceding chapter on forestry. This example clearly demonstrates one of the main problems with hand-engineered image interpretation systems. Such systems tend to perform adequately only within a narrow range of operating conditions, atypical of real world scenarios. In response to this and other aforementioned problems, various *automated* ways of constructing image interpretation systems have

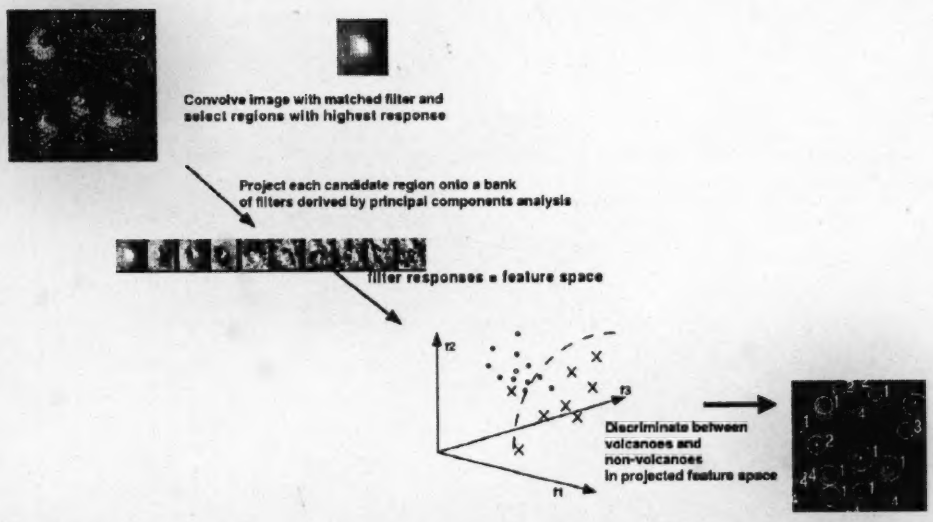


Figure 3.2: On-line testing phase of the volcano detection system [Burl et al., 1998]. First the previously created template is convolved, via a cross correlation procedure, with the input image generating regions of interest. The ROI's are then projected onto the subspace manifold created by the PCA procedure. The projection coefficients are used as input features to the classifier. The output is the classification of the ROI as volcano or non-volcano region.

been explored in the last three decades [Draper, 2003].

3.3 Learning Operator Parameters

Learning operator parameters is intimately related to the task of robust object recognition. Most vision operators found in literature have various controls, the settings of which strongly affect output quality. The optimal parameter values depend both on the task at hand and the input data. Furthermore, in most operators the parameter search space is prohibitively large and parameters interact in complex, non-linear ways, thereby making it practically impossible to explicitly model parameter behavior in a rule-based or algorithmic fashion. Thus, dependencies between input data and recognition goals in addition to complex parameter interactions create a need for dynamic parameter selection.

Initial attempts to optimize operator parameters using machine learning were presented in [Bhanu et al., 1995]. The authors used genetic algorithms to optimize parameters of the Phoenix color segmentation algorithm. The goal was to produce the best segmentation result that would be subsequently used by other subsystems for further processing. The multi-stage system consisted of 3 parts/stages: segmen-

tation, feature extraction, object recognition. The authors used a genetic learning algorithm [Goldberg, 1987] to obtain near-optimal parameter settings based on image content. This off-line component of the system was run on a number of training images to create a database of "good" parameter settings associated with extracted image features. The fitness function was composed of 5 different quality measures measuring both global and local segmentation properties. Two global properties: edge-border coincidence and boundary consistency, measure overall segmentation quality without explicit knowledge about the target objects. The three local measures: pixel classification, object contrast, and object overlap, require ground-truth about the objects present within the image. During the on-line "testing" phase, a 10-nearest neighbor component was used to select training exemplars closest to the given test image. The learned parameters were then used as initial seed points for the online version of the genetic algorithm, on order to bootstrap the search process with good initial starting points. This approach was able to achieve a 95.8% segmentation quality with respect to optimal[2] as compared to only 55.6% obtained by using default parameter settings and 63.2% obtained by using optimal parameters from a single training image.

The parameter selection task was once again tackled using a theoretically well grounded approach in [Peng & Bhanu, 1998]. Researchers used reinforcement learning to find optimal parameters for the same segmentation operator. The goal was to find a mapping between an input image (or its extracted features) and optimal parameter settings for the Phoenix segmentation operator. Once again the system described in the paper was composed of three stages, namely: segmentation, feature extraction, object recognition, with respective algorithm parameters being $[learned, fixed, fixed]$. The closed loop system (depicted in Figure 3.3) is based on obtaining immediate (associative) rewards back from the environment by feeding back the confidence score produced by the final/terminal object recognition procedure. The system uses teams or groups of (quasi-linear Bernoulli) units trained on encoded image features and the confidence score (feedback), in order to output operator parameters. In other words, the inputs to the stochastic neural net were the image features and the confidence score at time t, the resulting output was a new parameter set for time $t + 1$. The system looped until the confidence score (i.e., the quality measure) surpassed a predefined threshold or a predefined maximum number of iterations has been exceeded (implying that the object was not present or could not be found within the image). The experimental results seem

[2]Optimal segmentation parameters were obtained by exhaustively trying all possible parameter combinations made possible by the small search space for this specific experiment. The authors report that on average only 2.5%-5.0% of the whole search space was explored by the (on-line) genetic algorithm, which was able to converge to near optimal results in just three generations (using a population size of 10) during the testing phase (down from nine generations needed, on average, during the training phase).

to demonstrate the near optimal performance of the algorithm in contrast to the total segmentation failure produced by running the Phoenix algorithm with default parameters.

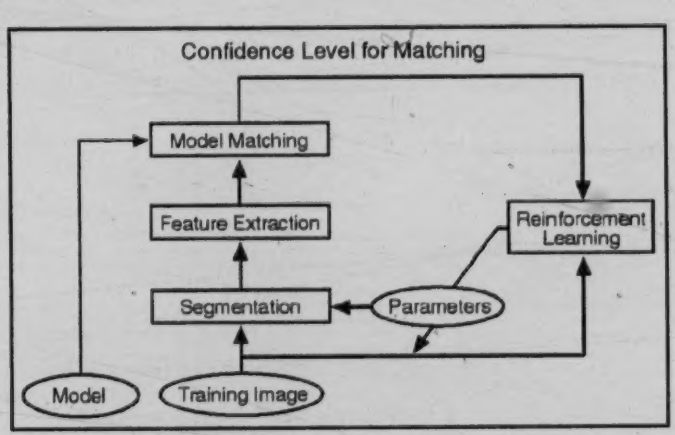


Figure 3.3: Object recognition system used in [Peng & Bhanu, 1998]

In [Bhanu & Peng, 2000] research within the two aforementioned directions was combined into a more complete system. Edge-border coincidence was used as immediate reward or *local feedback* to the parameter selection module (using the stochastic gradient descent described previously) of the initial segmentation stage of the system. The confidence score produced by the later object recognition procedure served as global feedback to the segmentation module. Initially local feedback was used to find good starting segmentation parameters. Once a predefined quality threshold has been reached the system started using global RL procedure as feedback into the segmentation algorithm. In a comparison to algorithm in [Peng & Bhanu, 1998], which used only the global feedback, the combination of local and global feedback allowed the system to converge on (i.e., learn) correct segmentation parameters in a significantly faster time.

Judging from the experimental results presented in the reviewed literature, closed-loop segmentation appears to be a promising technique. In fact edge-border coincidence and boundary consistency may be the right quality measures for evaluating the output of tree delineation algorithms. Thus, closed-loop valley-following may indeed be a plausible approach for automatically finding near-optimal parameter settings for individual tree delineation. In general, closed-loop systems need a feedback signal. Although the researchers found applicable feedback signals (namely edge-border coincidence and boundary consistency) for the task of color segmentation, it is not clear that such quality measures readily exist or can be easily created for find parameters of other image processing algorithms.

3.4 Learning Control Policies

3.4.1 Schema Learning System (SLS)

In response to the non-adaptive nature of systems using a single static sequence of operators, researchers have turned their attention to using multi-sequence systems. By adopting the notion of "goal-directed vision" [Draper et al., 1996], proponents of the theory argued that robustness and portability could only be achieved through the use of common vision operators (primitives) that are chained to form operator sequences, each designed for a special purpose task. Thus, autonomous systems creation lies with treating computer vision as a control problem over a space of image processing operators. Initial systems, such as the Schema system [Draper et al., 1989], which was named and based on Arbib's notion of Action Schemas [Arbib, 1978], had control policies consisting of *ad hoc*, hand-engineered rules. While enabling a systemic way of designing image interpretation systems, the approach required a large degree of human intervention. Noting the undesirable need for human-intervention, the schema learning system (SLS) was developed in [Draper, 1993] to remove the reliance on hand-engineered rule-base creation. The SLS system used recognition graphs to encode object recognition strategies that consisted of sequences of operator applications to intermediate data tokens. Exploration, learning from examples and optimization formed the three step process for creating recognition strategies. Off-line exploration applied all possible operator sequences to all training images, followed by the creation of an initial recognition graph. Graph optimization then pruned the initial recognition graph by removing sequences that did not produce goal-hypotheses or were redundant. Decision tree classifiers were also created during off-line training which enabled the system to decide whether to accept or reject intermediate level data tokens based on the discrete value ranges of the extracted features (see Figure 3.5). Accepted tokens were transformed by vision operators into more abstract tokens and again subjected to the accept/reject decision process. The procedure terminated when no more object recognition hypotheses were accepted by the final decision process (decision tree).

The SLS was the first system to attempt to learn special purpose strategies for object recognition. Unfortunately, it had several drawbacks. First, features could take on a relatively small number of discrete values, in order for decision trees to encode rules for selecting/rejecting data tokens. Second, the use of recognition graphs prohibited backtracking. Once the system made a decision it could not be "undone". As a result the system was essentially restricted to boolean decision making (accept data token or reject data token). Either a hypothesis was produced or the system signaled that no target object was located. In addition, no estimate of the quality of the hypothesis was possible.

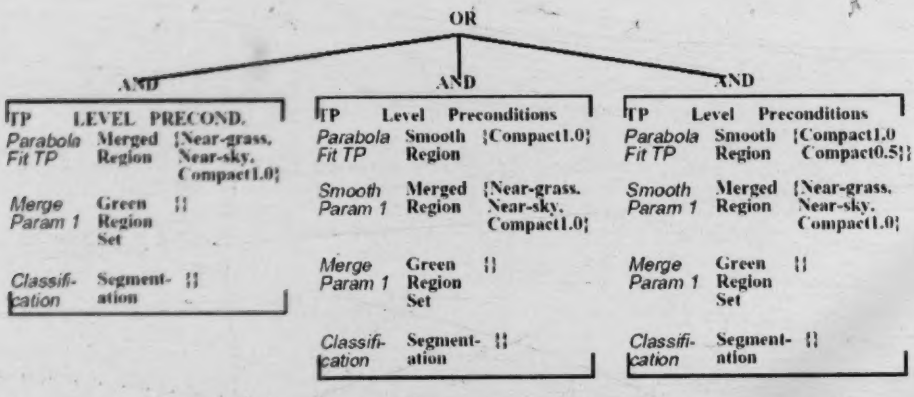


Figure 3.4: Recognition graph governing the application of vision procedures produced by the off-line component of the SLS [Draper, 1993]. Encoded as a DNF, the recognition graph represents a set of if-then rules where the antecedents are specific feature values and consequents are the visual procedures.

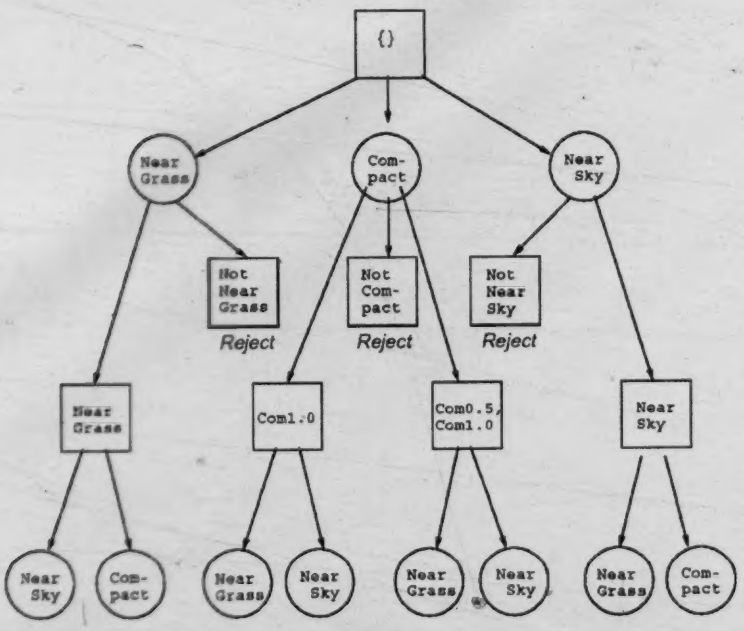


Figure 3.5: Decision Tree used for deciding to accept/reject a data token produced by the SLS [Draper, 1993].

3.4.2 Adaptive Object Recognition (ADORE)

In [Draper, 1996a], the SLS system was improved by changing the machine learning approach. Later renamed as ADaptive Object REcognition (ADORE), the system learned dynamic image interpretation strategies for finding buildings in aerial images [Draper et al., 2000]. As with many vision systems, including its predecessor, it identified objects (in this case buildings) in a multi-step process. Raw images were the initial input data, while image regions containing identified buildings constituted the final output of the system; in between the data could be represented as intensity images, probability images, edges, lines, or curves. The novelty was that ADORE modelled image interpretation as a Markov Decision Process, where the intermediate representations were continuous state spaces, and the vision procedures were actions. The goal was to learn a dynamic control policy that selects the next action (i.e., image processing operator) at each step so as to maximize the quality of the final image interpretation.

As a pioneering system, ADORE proved that a machine learned control policy was much more adaptive that its hand-engineered counterparts by outperforming any hand-crafted sequence of operators within its library. As such, it became the first system to **learn** a complex, domain and object specific control policy for object recognition. In addition, the system was effortlessly ported to recognize stationary (staplers, white-out, etc.) in office scenes and again was shown to outperform operator sequences designed by human domain experts [Draper et al., 2001].

In contrast to the original SLS system, ADORE used artificial neural networks [Haykin, 1994] to approximate the Q-value of a given state-action pair, and therefore could accommodate continuous feature spaces. Because ADORE selected each state-action pair in a greedy fashion, it was able to backtrack and select the state-action pair with the largest expected future reward. Finally, the approximate Q-value of the final hypothesis produced by the system, could be interpreted as an estimate of quality for the produced hypothesis. Unlike the system presented in [Isukapalli & Greiner, 2003], ADORE and its predecessor SLS, are designed to use interdependent operators, where the input to one operator is the output of another operator. On the other hand, the system presented in [Isukapalli & Greiner, 2003] requires operators that can be applied in any order. However, from a larger perspective the systems are similar in the sense that both use off-line dynamic programming to learn a guidance function used by the on-line module for selecting promising state-action pairs.

3.5 Summary of Related Work

To date research within computer vision has concentrated on development of specific subcomponents. In contrast, research on creating *complete* vision systems has been very limited. Clearly, designing a complete system is a challenging endeavor, regardless of the approach taken. Performance of the whole system depends on the performance of sub-components. Performance of the sub-components, in turn, is dependant on operator parameters. Optimal parameter settings, in turn, are dependant on the input image. While methodologies for analysis and optimization of systems and individual components have been developed, for example in [Greiffenhagen et al., 2001], such a process is difficult and labor intensive due to the complex interdependencies between components. As a result knowledge-engineered systems have a long development cycle, are expensive to maintain and are difficult to port to other domains. To automate the system creation process, researchers have turned to machine learning methods aimed at (i) learning efficient control policies to guide the operation of sub-components and (ii) selecting parameters for the sub-components in order to optimize the performance of the over all system. This chapter reviewed some of the most prominent, and in the opinion of the author, most promising techniques for automatic construction and optimization of vision systems.

Chapter 4

The Framework and Novel Contributions of MR ADORE

4.1 Introduction

The purpose of this chapter is to explain the design and operation of MR ADORE. In addition, the chapter highlights the novel contributions, in the form of framework extensions, that take the state-of-the-art in object recognition systems one step closer to being fully autonomous.

Figure 4.1: An example of a training image and the corresponding labeling (ground-truth) produced by a human domain expert. The pixels containing pure green color correspond to the pixels belonging to conifer trees, and the pure blue pixels correspond to pixels belonging to the shadow class. The hypotheses produced by the system can be evaluated against the ground-truth image to asses the quality of each hypothesis.

4.1.1 MR ADORE Design Objectives

MR ADORE was designed with the following objectives as its target: (i) rapid system development for a wide class of image interpretation domains; (ii) low demands on subject matter, computer vision, and AI expertise on the part of the developers; (iii) accelerated domain portability, system upgrades, and maintenance; (iv) adaptive image interpretation wherein the system adjusts its operation dynamically to a given image; and (v) user-controlled trade-offs between recognition accuracy and utilized resources (e.g., run-time).

These objectives favor the use of readily available off-the-shelf image processing operator libraries (IPLs). However, the domain independence of such libraries requires an intelligent policy to control the application of library operators. Operation of such control policy is a complex and adaptive process. It is *complex* in that there is rarely a one-step mapping from input images to final interpretation; instead, a series of operator applications are required to bridge the gap between raw pixels

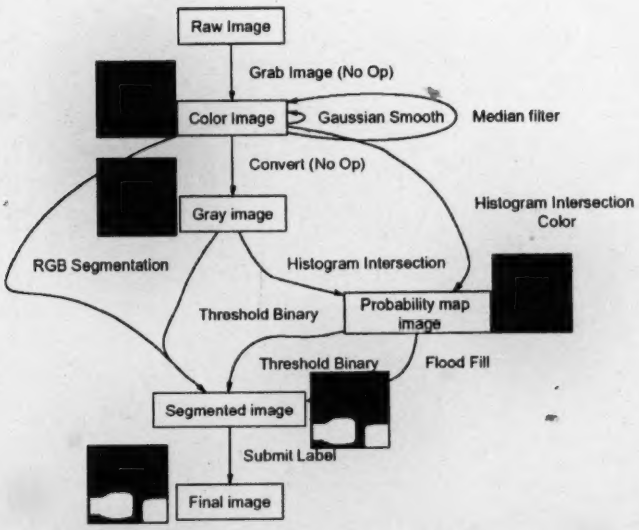


Figure 4.2: A reproduction of Figure 1.2 for the reader's convenience. Partial operator graph for the domain of forest image interpretation. The nodes and the corresponding example images depict that data processing layers, which in turn describe the *type* of MDP states present with MR ADORE. The edges represent vision routines, typically ported from the Intel OpenCV and IPL libraries, that transform one state to another (i.e., the MDP actions).

and semantic objects. Examples of the operators include region segmentation, texture filters, and the construction of 3D depth maps. Figure 4.2 presents a partial IPL operator dependency graph for the forestry domain.

Image interpretation is an *adaptive* process in the sense that there is no fixed sequence of actions that will work well for most images. For instance, the steps required to locate and identify isolated trees are different from the steps required to find connected stands of trees. Figure 4.3 demonstrates two specific forestry images that require different operator sequences for satisfactory interpretation results. The success of adaptive image interpretation systems therefore depends on the solution to the control problem: for a given image, what sequence of operator applications will most effectively and reliably interpret the image?

4.2 MR ADORE Operation

MR ADORE starts with a Markov decision process (MDP) [Sutton & Barto, 2000] as the basic mathematical model by casting the IPL operators as MDP **actions** and the results of their applications (i.e., data tokens) as MDP **states**. However, in the

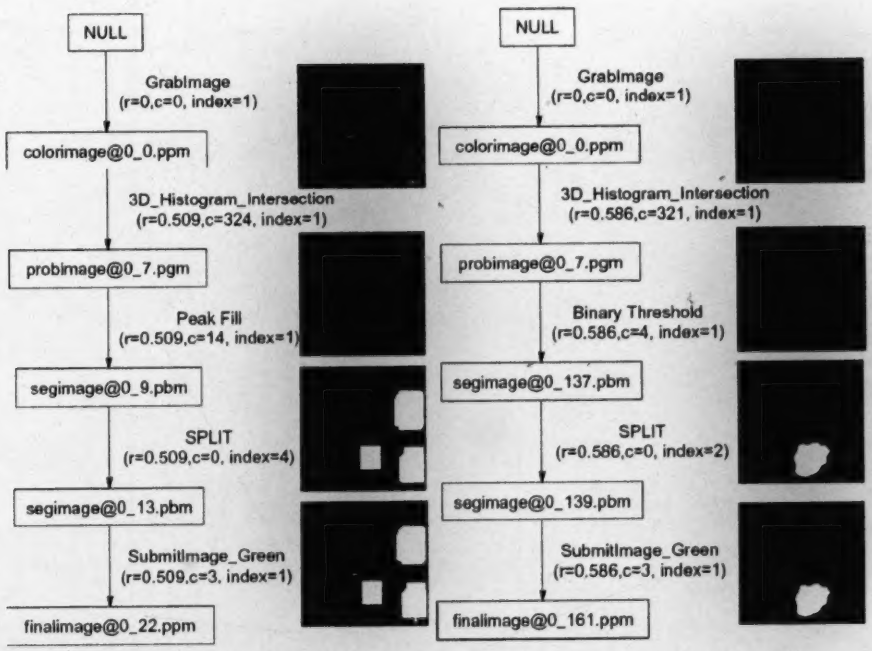


Figure 4.3: Adaptive nature of image recognition: two different input images require significantly different satisfactory operator sequences. Each node is labeled with its data type. Each arc between two data tokens is shown with the operator used.

context of image interpretation, the formulation frequently leads to the following challenges absent from typical search settings and standard MDP formulations:

1. Standard machine learning algorithms cannot learn from raw pixel level data since the individual states are on the order of several mega-bytes each. Selecting optimal features as state descriptions for sequential decision-making is a known challenge in itself.

2. The number of allowed starting states (i.e., the initial high-resolution images) alone is effectively unlimited for practical purposes. Additionally, certain intermediate states (e.g., probability maps) have a continuous nature.

3. In addition to the relatively high branching factor possible, due to large image processing operator libraries, some of the more complex operators may require hours of computation time.

4. Unlike standard search, typically used to find the shortest sequence leading to a goal state, MR ADORE attempts to find/produce the best image interpretation. In that respect the system solves an optimization problem rather than

one of heuristic search. Therefore, since goal states are not easily recognizable as the target image interpretation is usually not known *a priori*, standard search techniques (e.g., IDA* [Korf, 1990]) are inapplicable.

In response to these challenges MR ADORE employs the following off-line and on-line machine learning techniques. First, the domain expertise is encoded in the form of training data. Each training datum consists of two images, the input image, and its user-annotated counterpart, which enables the output of the system to be compared to the desired image labeling (typically called ground-truth). Figure 4.1 demonstrates a training pair for the forestry image interpretation domain. Second, during the off-line stage the state space is explored via limited depth expansions of all training image pairs. Within a single expansion all sequences of IPL operators up to a certain user-controlled length are applied to a training image. Since training images are user-annotated with the desired output, terminal rewards can be computed based on the difference between the produced labeling and the desired labeling. System **rewards** are thus defined by creating a scoring metric that evaluates the quality of the final image interpretation with respect to the desired (used provided) interpretation. Dynamic programming methods [Barto et al., 1995] are then used to compute the value function for the explored parts of the state space. The value function is represented as $Q : S \times A \rightarrow \mathbb{R}$ where S is the set of states and A is the set of actions (operators). The true $Q(s, a)$ computes the maximum cumulative reward the policy can expect to collect by taking action a in state s and acting optimally thereafter. See Figure 4.4 for an illustration of the process.

Features (f), used as **observations** by the on-line system component, represent relevant attributes extracted from the unmanageably large states (i.e., data tokens). Features make supervised machine learning methods practically feasible, which in-turn are needed to extrapolate the sampled Q-values (computed by dynamic programming on the explored fraction of the state space) onto the entire space (see Figure 4.5).

Finally, when presented with a novel input image, MR ADORE exploits the machine-learned heuristic value function $Q(f(s), a)$ over the abstracted state space, f(S), in order to intelligently select operators from the IPL. The process terminates when the policy executes the action `Submit(`$\langle labeling \rangle$`)`, which becomes the final output of the system (see Figure 4.6).

4.2.1 Building an Operator Library (MDP Actions)

Previous versions of ADORE used either hand-crafted vision operators or vision routines ported from the Khoros library. Since the Khoros library is no longer supported we turned our attention to other image processing frameworks. Intel provides two such packages. First the Image Processing Library (IPL) provides the

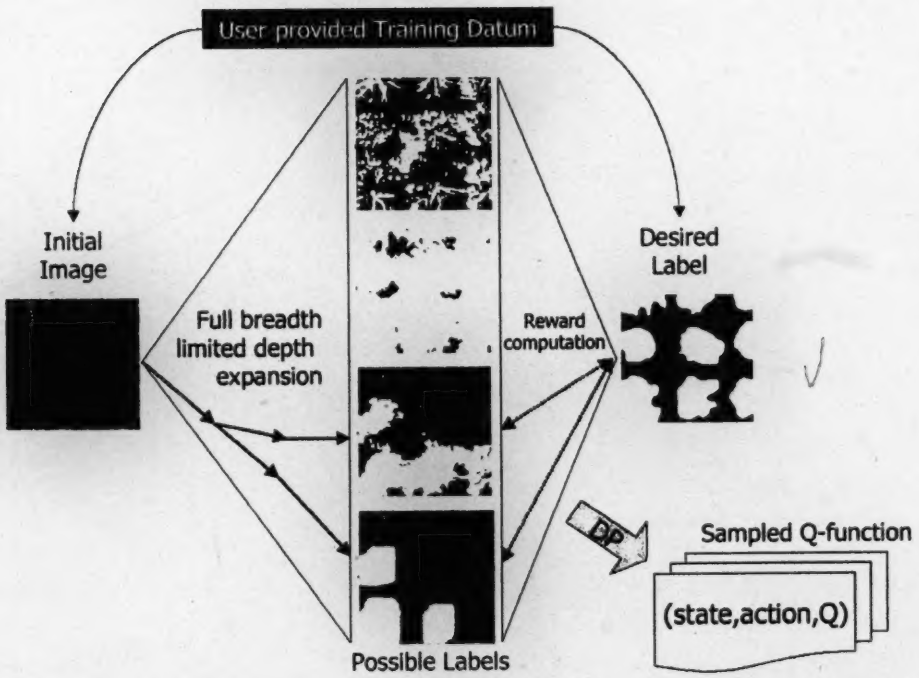


Figure 4.4: Off-line operation. First, full breadth limited depth expansion of each training image is performed producing a number of hypotheses about the locations of targer objects within the image. Each hypothesis is evaluated against the user annotated interpretation of the image (ground-truth) producing a quality measure (i.e., reward) for each system produced hypothesis. The rewards are propagated back up the tree producing a Q-value for each state-action pair.

image data structure as well as primitive operations (such as image creation, deletion, addition, filtering algorithms - Gaussian/Median smoothing). In addition to the IPL library, the OpenCV library provides more sophisticated computer vision algorithms such as edge detection, contour extraction and morphological operators. The OpenCV library is built on top of the IPL primitives. Despite the compatibility of the two libraries, creating the initial set of vision operators comprising a library for ADORE was not a quick nor an easy task. In order to create an adequate library, two undergraduate summer students and along with the author spent one full summer (four months) developing operators for ADORE. Following the summer development drive, the author continued developing operators for an additional four months. A subset of operators used by MR ADORE system is described in detail in Appendix A.

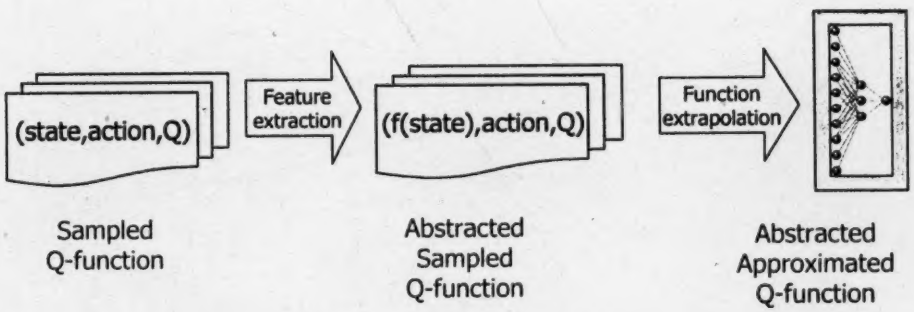


Figure 4.5: Feature extraction and Q-function approximation.

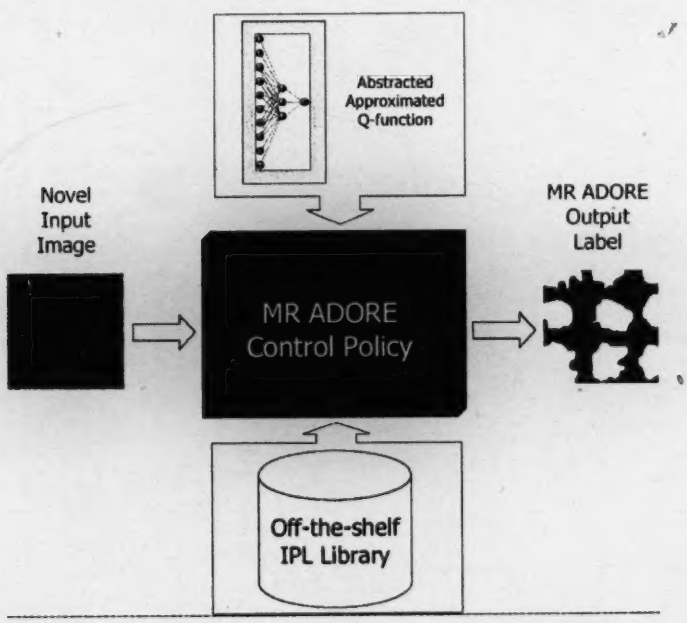


Figure 4.6: On-line operation. Using the machine-learned Q-function approximation, MR ADORE executes actions on states with the highest state-action Q-value. The process is terminated when a hypothesis deemed best by the system is Submitted to the user as the final interpretation.

4.2.2 System Rewards

In order to evaluate the quality of a given hypothesis (see Figure 4.4) a scoring metric is required. Consider the Venn diagram in Figure 4.7 where set A represents

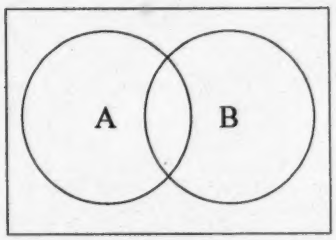


Figure 4.7: A typical Venn diagram depicting two sets A and B. In our case set A represents the set of pixels corresponding to the ground-truth. Set B corresponds to the set of pixels submitted by a sequence of operators (i.e., one specific hypothesis)

all pixels of the target object class (i.e., ground-truth) and set B corresponds to a set of pixels produced by a hypothesis (as a result of applying a specific operator sequence) for the same object class. We can then define a Reward function for a given hypothesis h (i.e., a sequence of operators) as follows:

$$\tilde{R(h)} = \begin{cases} 1 & \text{if } |A| = 0 \text{ and } |B| = 0 \\ \frac{|A \cap B|}{|A \cup B|} & \text{otherwise} \end{cases} \tag{4.1}$$

where $|\cdot|$ represents the cardinality of a given set. Commonly used in vision research, this pixel-based scoring metric computes the overlap between the set of hypothesis pixels produced by the system B and the set of pixels within the ground-truth image A. If sets A and B are identical then their intersection is equal to their union and the score/reward is 1. As the two sets become more and more disjoint the reward decreases, indicating that the produced hypothesis corresponds poorly to the ground-truth, set A.

Notice the first case in equation 4.1 applies when no target object is present within an image (i.e., $|A| = 0$) and the system submitted an empty hypothesis (i.e., $|B| = 0$) producing the highest reward of 1. In contrast, if the system submits anything other than the empty hypothesis for the case when no target objects are actually present in the image, the reward is 0. Thus, in this special case of $|A| = 0$, rewards are Boolean. In the case when $|A| > 0$, indicating the presence of target object(s) within the image, rewards become real numbers within the range of [0,1]. The second case depends on the overlap of ground-truth with the submitted hypothesis. If $A = B$ then $A \cap B = A \cup B$ and the reward is one. If, on the other hand $A \cap B = 0$, implying no pixels submitted correspond to pixels within the set of ground-truth pixels then the reward is 0. The values in between depend on sizes of sets A and B as well as the degree of overlap between the two. Thus, Equation 4.1 effectively combines the two commonly used concepts, precision and recall.

Precision Defined as:

$$\frac{|A \cap B|}{|B|} \tag{4.2}$$

precision measures the number of pixels identified correctly over the number of pixels classified as members of the target class. Hence precision score is maximal when number of *false positives* is minimal.

Recall Defined as

$$\frac{|A \cap B|}{|A|} \tag{4.3}$$

recall measures the number of pixels identified correctly over the number of pixels belonging to the target class. Hence recall score is maximal when the number of *false negatives* is minimal.

False Positives The pixels **incorrectly** identified as belonging to the target class.

False Negatives The pixels belonging to the target class that are **incorrectly** identified as non-target class pixels.

Since

$$\frac{|A \cap B|}{|A \cup B|} \leq \frac{|A \cap B|}{|B|} \tag{4.4}$$

and

$$\frac{|A \cap B|}{|A \cup B|} \leq \frac{|A \cap B|}{|A|} \tag{4.5}$$

implies

$$\frac{|A \cap B|}{|A \cup B|} \leq min(\frac{|A \cap B|}{|A|}, \frac{|A \cap B|}{|B|}) \tag{4.6}$$

Thus equation 4.1 can be viewed as a measure of both false positives and false negatives. As the score defined by equation 4.1 increases, precision or recall is increasing. Thus by combining precision and recall together into equation 4.1, both false positives and false negatives are penalized.

4.2.3 Features as Observations

During the off-line training phase the system is allowed access to the rewards computed by equation 4.1. More specifically, the off-line learning phase tries to learn a mapping $\langle f(s), a \rangle \longmapsto \mathbb{R}$, where s is a state (data token), a is an action and $f(\cdot)$

is a feature extraction function. As previously mentioned, modern machine learning methods cannot cope with raw images consisting of thousand or even millions of pixels. Thus, in order to create a function approximator via machine learning methods, image features must be extracted. Since the quality of the approximation depends on the relationship between features and rewards, in addition to the need for descriptive features, the structure of rewards must also be taken into account. For instance, if the reward schema described in the last section is used then the features at the hypothesis evaluation layer must be structured in such a way so as to account for the precision/recall penalties assigned by the reward function (or equivalently the scoring metric). Recall that Equation 4.1 tries to minimize the number of false positives and false negatives and as an illustrative example consider the images within Figure 4.8. If features are only extracted from the masked area then clearly the reward approximator can at best learn to compensate for false positives but *not* for false negatives. Consider Image 4 in Figure 4.8 depicting the parts of the original image within the hypothesis mask (Image 3). Looking only at this part of the whole image one can tell which pixels within the masked area do not belong to the target class as depicted by overlaying the ground-truth image over the masked image (Image 5). The non-green pixels within Image 5 show all the false positives within the hypothesis mask. On the other hand, looking at Image 6 in Figure 4.8 can tell us the pixels that actually belong to the target class but were missed by the current hypothesis. Thus in order to properly evaluate the quality of (or approximate reward for) a given hypothesis, features from both: the area within the hypothesis mask and the area outside the hypothesis mask must be extracted.

Before describing approximators capable of accounting for both false positives and false negatives additional terminology is in order.

I - an input image. (e.g., Image 1 in Figure 4.8).

M - a binary hypothesis mask. (e.g., Image 3 in Figure 4.8).

IM - image I multiplied by mask M. (e.g., Image 4 in Figure 4.8).

$I\bar{M}$ - image I multiplied by the inverse of mask M. (e.g., Image 6 in Figure 4.8)

Inputs for a Reward approximator

As shown in the previous section, features must be extracted from both IM and $I\bar{M}$ in order to have an approximator capable of accounting for both false positives and false negatives. So first features $f_{IM} = f(IM)$ and $f_{I\bar{M}} = f(I\bar{M})$ are created. The simplest way to combine these features is to concatenate the two vectors together ($f_{IM+I\bar{M}}$) and use the new feature vector as input into the function approximator. Here the hope is that the chosen machine learning algorithm is capable of learning to penalize for both false negatives and false positives.

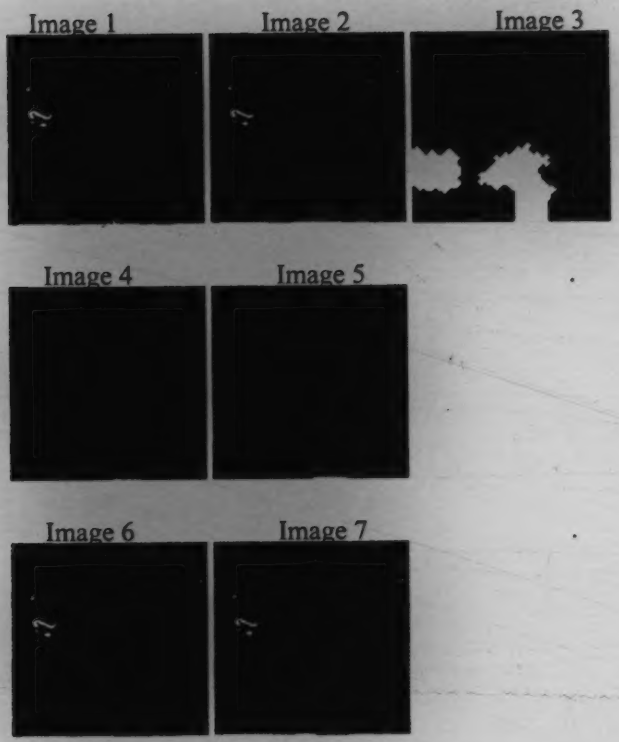


Image 1 Original Image.
Image 2 ground-truth Labelling. **Green** - conifer tree class. **Blue** - Shadow class. **Red** (not present) - deciduous tree class. **Unlabelled** - ground and other objects.
Image 3 Hypothesis mask for the conifer class derived by applying a sequence of operators.
Image 4 Image area within the hypothesis mask.
Image 5 Ground-truth labelling for pixels within the mask.
Image 6 Image area outside the hypothesis mask.
Image 7 Ground-truth labelling for pixels outside the mask.

Figure 4.8: Calculation of false positives and false negatives. In Image 5, the area under the hypothesis mask contains the target class pixels (green) and the false positives (non-green pixels). Notice that false negatives cannot be deduced strictly from this image. In Image 7, the green pixels represent the false negatives. By extracting features from both Image 4 and Image 6, the approximator is capable of learning to penalize a hypothesis for both false positives and false negatives.

A more direct approach would be to explicitly compose rewards. By presenting f_{IM} and $f_{I\bar{M}}$ to the corresponding precision/recall approximators, independent approximations of precision and recall scores can be obtained. Consider

$R_{IM} = R(f_{IM})$. Here we have the reward (assuming the features and the machine learning algorithm produce an adequate approximation) based on *recall* as false negatives cannot be accounted for. On the other hand $R_{I\bar{M}} = R(f_{I\bar{M}})$ is the reward for submitting the opposite hypothesis (i.e., all pixels outside the initial hypothesis mask). By subtracting $R_{I\bar{M}}$ from R_{IM} a simple approximation to the real evaluation formula can be obtained.

The approach implemented in MR ADORE, however, differs from both of the aforementioned feature extraction methods. The simplest possible approach is adopted by concatenating f_I together with f_{IM} and $f_{I\bar{M}}$ to produce $f_{I+IM+I\bar{M}}$. This appears to be the most general approach requiring the least amount of vision and machine learning expertise and lends it self well to the overall goal of full system automation.

4.3 Extending the Framework

The overall goal of this research is to create a system that can produce accurate image interpretations in an efficient manner. In order to realize this goal the following framework extensions have been designed:

- Add efficient pruning methods. Clearly MR ADORE is a complex system that requires a vast amount of computational resources. Although the power of modern computers is growing rapidly, optimizing the systems performance is nevertheless a crucial step in order for the approach to be feasible for real world applications. However, the original framework of ADORE had no optimizations of any kind. Because the system relied heavily on search techniques, methods that reduce the combinatorial state space can greatly reduce the runtime of the system. Thus, one of the research directions concentrated on developing effective pruning techniques that remove inadmissible and redundant states from the search space, thereby optimizing the performance of the system. The research is presented in section 4.3.1.

- Reduce the need for control points within MR ADORE. Original ADORE made decisions whether to apply an action (vision routine) to a given token. While this follows the standard reinforcement learning approach, the method requires building a control point (i.e., train a function approximator) for every possible action. In addition, each data processing level requires a set of relevant features. Feature extraction, in turn, requires a domain/vision expert. Thus while ADORE enabled the use of a generic operator library, feature selection remained one of the last vestiges of human intervention within the Adaptive Object Recognition framework. However, as the experimental results in the next chapter will show, MR ADORE can outperform any given

static policy produced by a sequence of operators within the library, while needing only a single set of features and only one corresponding machine-learned function approximator. The research is presented in section 4.3.2.

- Removing the dependency on hand crafted features altogether. While previous versions of ADORE can adaptively select operator sequences, the systems still required hand-crafted features created by domain experts. Thus, the most ambitious research goal of this thesis was to completely remove the last vestiges of human intervention within the Adaptive Object Recognition framework by enabling automatic feature extraction within MR ADORE. The research is presented in section 4.3.3.

- Apply the new system to a natural (as opposed to a man-made) domain. One of the fundamental goals of this research project was to create a system for the forestry domain. While the overall goal of the project remains individual tree parameter extraction, the current phase of the project is focused on using the MR ADORE system to segment images of forest scenes. Unlike the research in [Bhanu & Peng, 2000], where optimal parameters to an *unsupervised* segmentation algorithm are sought, our goal is to use MR ADORE as a *supervised* segmentation algorithm. Recall the model-based and image-based approaches discussed in chapter 2 and the large overlap of procedures used by both. By adapting the notion of "goal-directed" vision to the task of identifying individual trees within a scene, the following sequence of procedures forms a plausible solution approach. First image regions containing target objects are identified. This stage conforms to the task of supervised segmentation. Second individual tree delineation is performed followed by spatial conflict resolution. To that extent the system can be viewed as the first phase of the outlined approach to identifying individual trees of specific species within a scene. The overall process can be viewed as three applications of MR ADORE, each using a different operator library and taking different inputs. Chapter 5 presents experimental results conducted on both synthetic and real images of forest scenes.

4.3.1 State space pruning techniques

Because search is used in both off-line training phase and on-line policy execution phase, combinatorial increase in storage requirements is inevitable. During the off-line training stage all possible operator sequences up to a certain, user specified depth, are applied to a set of training images. However, the original ADORE applied operators regardless of the final state produced (i.e., leaves of the search tree) by a sequence of operators. Unfortunately, unless the sequence ends in a "terminal

state"[1] there is no way to evaluate the quality of the interpretation.

Consider the following airplane analogy. An airplane contains a certain amount of fuel f (i.e., user specified maximal depth of the search). Starting from point S (initial data token) the airplane needs to get to a terminal state T, the airport, in order to land. Suppose the airplane flies from point S to S' expending g units of fuel and suppose the closest airport (i.e., terminal state T) takes h units of fuel to get to from S'. Then if $g + h > f$ the airplane will crash before making it to the airport.

In a similar fashion, if a hypothesis cannot be produced by a sequence of operators, there is, in fact no interpretation produced. Thus sequences not ending in a terminal state within a user specified number of steps (i.e., constrained by maximal number of vision operators chained into a sequence) simply waste resources. Thus, the original ADORE wasted time and resources by executing vision operators, and producing data tokens that could not be evaluated. To optimize the effectiveness of the search, state pruning techniques are required. Two such pruning techniques have been devised for MR ADORE in order to eliminate production of redundant and inadmissible data tokes. (Table 5.1 shows the results of using pruning techniques to optimize the performance of the MR ADORE system.)

Logic Based Pruning

As mentioned previously, if a sequence does not end in a terminal state within the maximal number of steps it should not be generated. In order to reduce the combinatorial explosion of states, inadmissible states should, naturally, be pruned first. Logic based pruning is based on knowledge about the shortest sequence possible to a goal state given a library of vision operators. Like the airplane scenario, logic based pruning, looks at number of actions taken thus far [2] $g(s)$, and the minimal number of actions to a terminal state $h(s)$. If total number of actions exceeds the user specified maximum no further actions are applied to the data token. So if $g(s) + h(s) > f$, all states resulting from token s are pruned. A minimum spanning tree [Cormen et al., 1990], rooted at the finalimage node easily captures all the necessary information needed to efficiently compute $g(s)$ for any state. In turn, Prim's and Kruskal's algorithms [Cormen et al., 1990] can easily compute a minimum spanning tree enabling the automation of this pruning technique.

[1] Terminal state is "finalimage" data token produced by applying the "Submit(Label)" procedure (cf. Figure 4.2)

[2] For our experiments all action costs were set to 1 unit within MR ADORE.

Content Based Pruning

Original ADORE implementation conducted a depth first search of the state space governed by the possible actions (i.e., vision operator library). However, if two sequences produce the same data token, one of the identical tokens should be removed. Content based pruning is designed to do exactly that (and a little more). This technique brings together two seemingly distinct research topics, namely $A*$-search style pruning and content based image retrieval.

4.3.2 Learning Control Policies

The purpose of the off-line learning phase within MR ADORE is to automatically construct the on-line control policy. The policies studied to date can be conceptually represented via the following unifying framework.

First, data tokens (i.e., states) can be split into disjoint "processing levels". The initial input image is a data token of the top (first) data level and the final submitted image interpretation is a data token from the bottom (last) data level. Figure 4.2 illustrates six processing levels as follows: *raw image, color image, gray image, probability map image, segmented image,* and *final image.* Consequently, the set S of data tokens can be partitioned into subsets $S_1, \ldots, S_n$ where S_i contains data tokens belonging to processing level i. Likewise, the set A of operators can be represented as a union of (disjoint) operator subsets A_i, where action $a_i \in A_i$ is applicable at level i.

Therefore, any control policy $\pi : S \rightarrow A$ can be conceptually divided into n control policies: $\pi_1, \ldots, \pi_n$ where π_i operates over S_i only, that is $\pi_i : S_i \rightarrow 2^{A_i}$. There are several different classes of π_i policies possible, including:

fixed: $\pi_i(s) = \{a_i\}$ for all states s. Such policies blindly apply the same action regardless of the data token s at hand.

full expansion: $\pi_i(s) = A_i$: all applicable operators within the set A_i are executed.

value-based: $\pi_i(s) = \{\arg\max_{a \in A_i} \hat{Q}_i(s, a)\}$ where $\hat{Q}_i$ is a machine-learned approximator used in place of the optimal Q_i-function[3]. Such policies greedily select actions to be executed with the highest $\hat{Q}$-values. This policy is specified by the machine-learned approximator and the feature selection function.

In addition to the three aforementioned types of processing level policies, two special policy types are possible within MR ADORE that do not conform to the standard policy definition presented above.

[3]Typically optimal Q-values are denoted by Q^*, while machine learned approximations are denoted by Q^{ML}.

path-selector: $\pi_i(s) = a^i \in A_i, a^{i+1} \in A_{i+1}, \ldots, a^j \in A_j$: This policy selects a (sub)sequence of actions, the first of which, namely a^i, is executed and the rest are passed on to the next processing level(s). This policy is really a derivative of the value-based policy in the sense that it too is a Q-greedy policy defined by a machine-learned function approximator and the feature selection function. The most notable difference is unlike the value-based policy, the path-selector selects a (sub)sequence of actions to execute rather than a (sub)set of actions.

follow-through: This policy $\pi_j(s)$ executes action a_j specified by a path-selector policy at a higher processing level, π_i, where $1 \leq i < j$. For example, a path-selector may have selected the following sequence of operators: a) convert color image to grayscale image, b) threshold the grayscale image thereby producing a binary image , c) invert the binary image, d) submit white pixels as the hypothesis indicating image regions containing target objects. The path selector would execute action (a), while the follow-through policies would execute actions (b)-(d).

Although these two policies significantly differ from the three initial polices, all five policy types can be mixed and matched to form the global policy π. (One obvious exception is that a follow-through policy must be proceeded by a path-selection policy at a higher level.)
Thus, any policy π can be described as an n-ary vector of π_i, where n is the number of processing layers. Selecting the optimal combination of π_i's type and parameters is a difficult problem. Pre-ADORE systems have solved this problem by employing domain engineering techniques in order to design π, which virtually always calls for extensive trial-and-error experiments and substantial human expertise. In [Draper et al., 2000] and [Draper et al., 2001] researchers used hand-crafted features to implement the value-based policy at all data levels (commonly known as best-first/greedy policy) within ADORE. Research in the MR ADORE project casts policy selection as a challenging open machine learning problem with the elements of meta-learning. To that end, the next subsections outline the following three types of control policies that have been explored to date.

Static Policies

A **static** policy, π_{static}, uses a single sequence of operators and therefore consists of only fixed π_i's. Research within the field of computer vision has produced numerous systems using a static sequence of operators (e.g. [Burl et al., 1998]). As previously mentioned such systems require extensive trial-and-error experiments, domain expertise, etc. Since ADORE and MR ADORE perform a full-breadth

search during the off-line phase, the **best** static sequence can be automatically determined from the off-line expansions of training images. Such policies, while being computationally inexpensive (on-line), generally can only achieve near-optimal performance within a narrow range of operating conditions (due to the non-adaptive nature of the policy). In fact, this is the reason why researchers initially turned to using a library of operators and adopted the "goal-directed" paradigm for object recognition.

Trunk-based policies

A **"trunk-based" policy** π_{trunk} makes its only decision at the top processing level (S_1) by selecting an n-long sequence of operators based on the input image only. The policy maintains a library of *optimal*[4] sequences (called "trunks") collected during the off-line stage. Therefore, π_1 is a path-selector over the space of trunks while $\pi_2, \ldots, \pi_n$ are all of the follow-through type.

Least-commitment Policies

While best-first policies are theoretically capable of much more flexibility than static or (semi-static) trunk-based policies, they depend crucially on (i) data token features for *all* levels and (ii) adequate amounts of training data to train the Q-functions for *all* levels. Experience has shown that it is substantially harder to select features at the earlier levels where the data tokens exhibit less structure [Draper, 2003]. To make the problem worse, a single user-labeled training image delivers exponentially larger numbers of $\langle$ state, action, reward $\rangle$ tuples at later processing levels. Unfortunately, the first processing level gets the mere $|A_1|$ tuples per training image since there is only one data token (the input image itself). As a net result, best-first control policies have been known to backtrack frequently [Draper et al., 2000] as well as produce highly suboptimal interpretations [Bulitko & Levner, 2003], due to poor decision making at the top processing layers. In fact the trunk-based policy suffers from the same phenomenon, a lack of high-quality features and sparseness of training examples.

Rather than making control decisions at every level based on the frequently incomplete information provided by imperfect features, the **least-commitment policies** postpone their decisions until more structured and refined data tokens are derived. That is: $\pi_j = A_j$ for $1 \leq j \leq n-1$ and π_n is value-based. In other words, we apply all operator sequences up to a predefined depth and only then engage the machine-learned control policy to select the appropriate action. Doing so allows the control system to make decisions based on high-quality informative features, result-

[4] An optimal sequence is the one yielding the greatest reward over all other sequences with respect to a given input image.

ing in an overall increase decision quality. As a side benefit, the machine learning process is greatly simplified since feature selection and value function approximation are performed for considerably fewer processing levels, while benefiting from the largest amount of training data. (Figure 4.9 illustrates the outlined policies.)

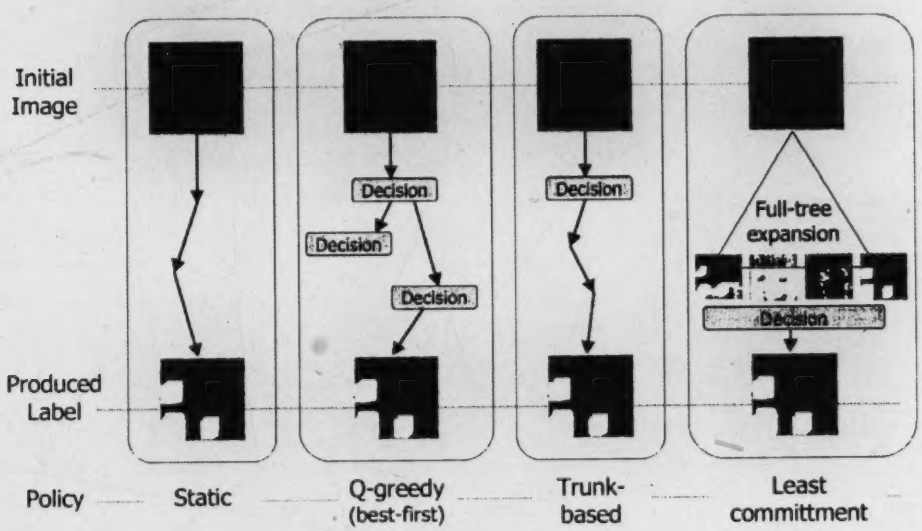


Figure 4.9: Four types of on-line polices.

4.3.3 Automatic Feature Extraction

Research into feature extraction has spanned decades. Because many machine learning algorithms cannot cope with millions of pixels, informative features are essential to building useful AI systems. Unfortunately, feature extraction is a tedious process requiring extensive domain expertise and algorithmic proficiency. As a result, automatic feature extraction procedures are highly desirable.

Raw Pixels and KNN

Contrary to the statement in the previous section, one algorithm *can* use raw pixels as features. The k-nearest-neighbors (KNN) algorithm [Duda & Hart, 1973] is a case-based reasoning algorithm that simply stores seen examples in memory. When presented with a new input, the "lazy-learner" finds training examples that resemble the new input the closest, and reasons based on stored information for the k-nearest neighbors. In the case of approximating reward, the goal is to find the hypothesis closely resembling the input hypothesis and reason based on the reward achieved

by the nearest-neighbor. Thus, feature extraction can be avoided all together by using raw pixels as features and employing the KNN algorithm to approximate the reward of a hypothesis.

Principal Component Analysis (PCA)

A commonly used method for image compression within the vision community is Principle Component Analysis (PCA). In a nutshell, Principal Component Analysis (also known as Karhunen-Loeve Transform, and Hotelling Transform)[Sirovich & Kirby, 1987; Kirby, 2001] computes the average image $I_{ave} = \frac{1}{n}\sum_i^n I_i$ from a number (n) of "training" images and then computes the variance $\triangle I = |I_{ave} - I_i|$, $\forall n$. Subsequently all image variances are used to create a matrix $A = [I_1, ..., I_n]$ and the covariance matrix is then created AA^T. Once the covariance matrix is constructed its eigenvalues (λ_j) and corresponding eigenvectors (x_{λ_j}) are computed (recall $Ax = \lambda x$), creating an image compression basis $\beta = [x_{\lambda_1}, ..., x_{\lambda_k}], (k < n)$, which can be used to compress the training and testing images. For any image I_α we calculate its variance $\triangle I_\alpha$ from the previously computed I_{ave} and then obtain a dot product value with each of the basis components $[\triangle I_\alpha \cdot x_{\lambda_1}, ..., \triangle I_\alpha \cdot x_{\lambda_k}] = cv$. Thus a feature vector (cv) of compressed values is obtained. Subsequently these feature vectors can be used to compute distances between training and input samples or be used by function approximators to compute Q-values.

4.4 Summary of Research Contributions

The research goal of this thesis is to push the state-of-the-art image interpretation systems one step closer to being fully autonomous. To solve this goal, this chapter presented the framework of MR ADORE and outlined the novel extensions. The next chapter presents experimental results supporting the applicability of ideas developed in this chapter.

Chapter 5

Experiments using MR ADORE

5.1 Introduction

Previous chapters of this thesis have: presented the domain of forestry, reviewed research regarding adaptive vision systems and presented the MR ADORE system which extends the framework of ADORE. One of the contributions of this thesis is to bring the emerging technology of Adaptive Object Recognition to the forestry domain, using the system for the first time to recognize natural (as opposed to man-made) objects. More significantly, the aim of the research is to remove the last vestiges of human intervention in ADORE, namely the need for human expertise. Previous versions of ADORE learned recognition strategies from training samples – but only after a human expert had selected the features that describe each intermediate level of representation. This task is not only expertise-intensive but also domain-specific. Furthermore, the original ADORE had no pruning methods developed for efficient operation of the search algorithms. This chapter presents the experiments conducted using the modified system and shows the corresponding results. In addition, the supplementary experiments demonstrate the robustness of the MR ADORE system.

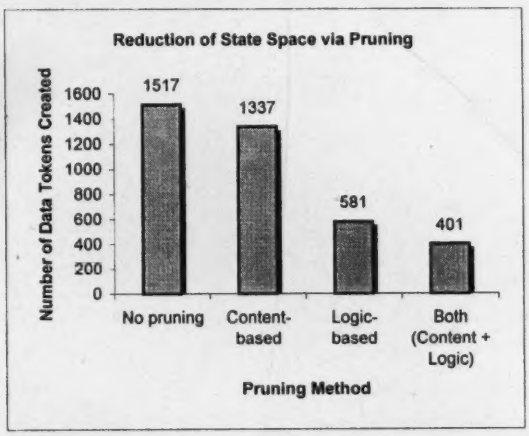
5.2 The Effects of State Space Pruning

As outlined in section 4.3.1, two pruning techniques have been designed: Logic based and Content based pruning. The effects of using each technique separately and in tandem are presented in Table 5.1. Clearly using both pruning techniques provides the most benefit.

5.3 On-line policies performance

In order to determine the merit of each policy presented in section 4.3.2, MR ADORE was tested with static, trunk-based and least-commitment policies along with two base-line policies: random and random trunking. A random policy is simply a least-commitment policy that randomly assigns rewards to data tokens rather than using a machine-learned function approximator. Similarly, a random trunk-based policy is a trunk-based policy that selects an operator sequence (i.e., a trunk) at random rather than using a machine-learned function approximator to assign Q-values to each of the possible sequences. As previously mentioned, trunk-based policies suffer from lack of informative features and training examples.

Table 5.1: Reduction in storage and time requirements as a result of using Content and Logic-based pruning techniques.



Pruning Method	# data tokens	time(min)
No pruning	1517	8.5
Content-based	1337	7
Logic-based	581	2
Both (Content + Logic)	401	<1

Setup

The performance evaluation was conducted on 35 images of forest plantations[1] (cf. Figure 4.1). To establish base-line performance, all possible operator sequences (constrained to a maximal length of 4 operators) were applied to all the images. The resulting **off-line optimal** scores are shown in Table B.2 of Appendix B along with other experimental details.

Experimental Results

Using a leave-one-out cross-validation strategy (i.e, 34 for training and 1 for testing), each of the five control policies was run on the set of forestry images. Table 5.2 shows experimental results for the trunk-based policies. These policies used a KNN based Q-function approximator in conjunction with various features and distance metrics. The best result of 76% was achieved using HSV histograms together with

[1]The images were acquired during the summer and fall of 2000. The plantation, located at the Vegreville, Alberta, Canada, contains young spruce and birch species surrounded by grassy meadows. We are grateful to Li Cheng, Terry Caelli and Ronghuo Man for providing us the images and their corresponding image interpretations.

Table 5.2: Performance of KNN trunk-based policies using different features and distance metrics (described in Appendix B). 35 images of forest plantation scenes were used in the experiment. Results for each policy were averaged over the 35 leave-one-out cross-validation runs. The performance of each policy is evaluated relative to the off-line optimal performance.

	L1	L2	Angle	RL1	RL2
lbp	0.66	0.69	0.45	0.66	0.69
HSV hist	0.76	0.73	0.69	0.76	0.73
RGB hist	0.72	0.72	0.73	0.72	0.72
RGB+HSV mean	0.68	0.69	0.71	0.69	0.7
HSV mean	0.72	0.72	0.7	0.71	0.72
RGB mean	0.59	0.62	0.23	0.59	0.62

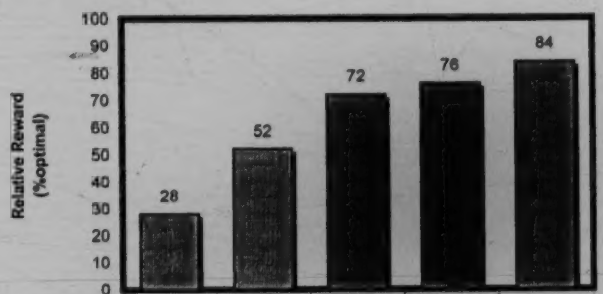
Table 5.3: Performance of SNoW and ANN least commitment policies using different features. 35 images of forest plantation scenes were used in the experiment. Results for each policy were averaged over the 35 leave-one-out cross-validation runs. The performance of each policy is evaluated relative to the off-line optimal performance.

	snow	ann
rgb hist	66	55
rgb mean	64	55
hsv hist	69	84
hsv mean	65	41
lbp	80	70

$L1$ distance metric (see Appendix B for a detailed description of distance metrics and feature used in these experiments). Table 5.3 shows experiments using the least commitment policies. In this set of experiments, neural networks and SNoW Q-function approximators were used. Each function approximator was tested using the features outlined in Appendix B, with the best result of 84% achieved by an artificial neural network using HSV histograms as input features. Finally, Table 5.4 shows a summary of the experimental results. The best results from the two previous tables are compared against the base line policies, namely static, random, and random trunking.

As expected, since the least-commitment policy uses a function approximator trained on data tokens at the bottom processing level, which contains exponentially more

Table 5.4: Performance of five different policies on the forestry data. 35 images of forest plantation scenes were used in the experiment. Results for each policy were averaged over the 35 leave-one-out cross-validation runs. **% optimal** refers to the fact that each policy is evaluated relative to the best off-line interpretation achievable given a particular IPL.



examples than preceding levels, this policy outperforms all others[2]. The trunk-based policy follows next[3]. As mentioned previously, the lack of information at the top processing layer clearly makes the trunk-based policy perform worse than the least-commitment policy. The results are consistent with previous work. In [Draper, 2003] the researchers informally report that as less and less informative features are used to train the best-first policy, it converges to a static policy (i.e., the best-first policy applies the same sequence of actions regardless of input). Thus, the performance of the static policy is almost as good as the best trunk-based policy since the trunk-based policy does not use informative features. Not surprisingly the random policies perform the worst. It should be noted that the chances of a policy randomly selecting an optimal sequence for every test image is virtually zero. Setting the maximum length for a sequence at four results in 118 possible sequences for each image and 35 images in total. For simplicity it is assumed that there is but one optimal sequence per image[4]. Thus $P(\pi^{optimal}) = (\frac{1}{118})^{35} \approx 3 * 10^{-72}$.

[2]The results presented here used an artificial neural network [Haykin, 1994] as the function approximator and an HSV histogram for input features. See Table 5.3 and Appendix B for more details.

[3]Here KNN was used with a number of features and distance metrics. The best results shown in the table were obtained by using HSV histogram features together with a Euclidean distance measure. See Table 5.2 and appendix B for more details.

[4]The author has observed two sequences being optimal for one image but only on very rare occasions. Never the less, the derived result is a lower bound for the probability of performing optimally by chance.

Analyzing the errors, made by the least-commitment policy, revealed that the worst performance occurs for images that do *not* contain any target class objects. During the course of the experiments a multitude of settings have been tried. In addition to using different features, a number of settings for the learning rate, momentum and hidden units were tried for neural networks and all produced the same result. In addition, SNoW expects a distributed input of features, so a number of different functions have been tried in conjunction with all the features. After all the trials **not a single configuration** was able to correctly interpret an image that contained zero target class pixels. When these images were removed from the test set, static, trunk-based, and least commitment policies all experienced a 5% increase in performance. It is the author's belief that the scoring metric defined by equation 4.1 is too strict in the case when the size of the target class pixel set is zero. Since the rewards are boolean for this specific case, all the least-commitment policies receive a reward of zero for the images that do not contain the target objects. While creating a more lenient scoring metric may be the immediate solution to the problem, focus of attention mechanisms, outlined in the next chapter as directions for future research, may eliminate this problem all together.

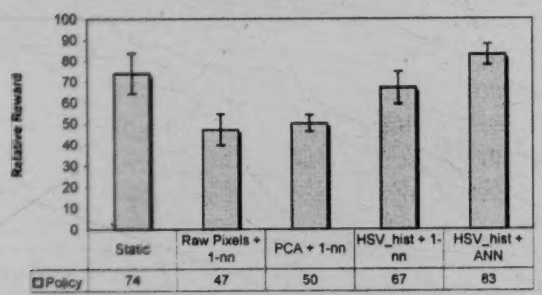
5.4 Automated Feature Extraction Performance

In order to test the two 'Automated' feature extraction algorithms, MR ADORE was run using KNN together with (i) Raw Pixels as features and (ii) PCA projection coefficients as features. Technical problems plagued this experiment from the start. First, the need to find the k-nearest neighbors is an on-line, CPU intensive procedure. If 34 training images would be used, the KNN procedure would need to compare every newly produced hypothesis to the corpora of stored training interpretations. Each training image at depth 4 produces 118 hypotheses resulting in the need to store $34 * 118 = 4012$ training examples at the bottom processing level. Since the test image also produces 118 hypotheses, every new hypothesis must be compared to every training hypotheses, resulting in $118 * 4012 = 473,416$ comparisons. Using raw pixel features implies that there is almost half a million image to image comparisons being done. Since the images used are 256×256 pixels, each comparison, regardless of the distance metric, needs at least $256 * 256 = 67,840$ pixel based operations. In total, to run one cross-validation run there are *at least* $67,840 * 473,416 = 32,116,541,440$ pixel based computations being performed. In reality, the scoring metrics[5] require significantly more operations to return a distance rendering the naive implementation of this approach somewhat impractical. In addition the PCA algorithm requires all training examples to be stored in memory.

[5]For this experiment L1, L2, Relative L1, Relative L2, distance metrics were used as described in Appendix B.

Therefore, physical memory size restricted the use of PCA to about 1000 images. That, in turn, implied that only 5 images could be used for training when PCA was employed as the feature extraction routine. Noting the two aforementioned difficulties 20 plantation images were used in a 4-fold cross-validation testing strategy, where each fold contained 5 training images and the other 15 images were used as test images. The experiment compared the performance of several policies. First a policy using the best static sequence of operators was evaluated using the aforementioned cross validation policy. Then policies using the two automated feature extraction algorithm were rum. Finally, two more policies were evaluated that used hand-crafted features from the previous experiment. On-line policies employing HSV histograms (see Appendix B.3 for more detail) respectively used 1-nn and ann reward approximation algorithms. The experimental results are shown in Table 5.5.

Table 5.5: Results of using PCA as a feature extraction algorithm compared to using raw pixels as features. Both approaches use k-nearest neighbors as the method to approximate the reward of hypothesis tokens. The performance of the static sequence is given as a base line comparison.



Clearly the algorithms using automated feature extraction do not perform nearly as well as the static sequence or algorithms using the hand-crafted features. Consequently, the small size of the training set is perhaps the first possible reason for the poor performance of the two algorithms for automated feature extraction. The second may be that the k-nearest neighbors algorithm is too naive in its assumptions to effectively approximate the Q-value of a given hypothesis. However, the policies which use hand-crafted features discredit both of these claims. The performance of the policy using artificial neural networks together with hand-crafted features did not significantly degrade when compared to the leave-one-out experiments presented in Table 5.4 of the previous section. Therefore the lack of training examples cannot attribute to the weak performance of policies using automated feature extraction algorithms. Similarly, the performance of 1-nearest neighbor algorithm increases by a statistically significant amount when hand-crafted features are used

in contrast to automated feature extraction algorithms. Conversely, if both policies using hand-crafted feature extraction algorithms are compared to each other, clearly the 1-nn algorithm performs statistically worse than the ANN algorithm. To summarize, the poor performance of policies using the automated feature extraction algorithms can be attributed mainly to the poor features extracted/used but also to the nearest neighbor algorithm, which cannot approximate the quality of a hypothesis as well as artificial neural networks.

From a practical point of view, a more efficient implementation of each algorithm (KNN and PCA) is necessary for future research. Incremental PCA methods have been proposed in [Hall et al., 2000] and need to be implemented in order for a larger training corpora to be used in the future experiments. In addition efficient KNN algorithms need to be researched and designed. Approaches such as finding approximate neighbors [Arya et al., 1998] or using kd-trees [Pelleg & Moore, 1999] as efficient search structures offer prospective starting points, although their direct use for pixel based KNN is not entirely obvious.

5.5 Supplementary Experiments

5.5.1 Labeling Errors Experiment

Substituting raw pixels or PCA projection coefficients in for features clearly needs much more research and development in order to make the approach feasible. However, during the analysis of the system's performance an interesting peculiarity was noticed. Shown in Figure 5.1, the KNN+PCA policy selected a result that visually looked better than both output of a static sequence *and* the expert labeled representation of ground-truth. Consistent with results presented in chapter 2, domain expert can and do make mistakes. In the case of Figure 5.1, a small pine tree was left unlabeled by the expert in row two, column two. However, MR ADORE was able to find it.

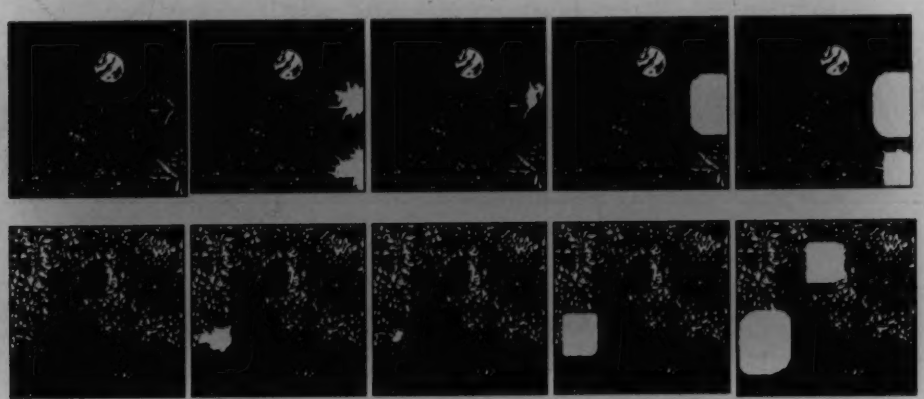


Figure 5.1: Each row from left to right: the original image, desired user-provided labeling, optimal off-line labeling, best static policy of length 4 labeling, best-first policy labeling (using KNN and automatically extracted features via PCA). The adaptive policy has been observed to outperform best static policy (top row) and sometimes the human experts as well (bottom row).

This observation naturally leads one to question the robustness of the system with respect to errors in the labeled images. In an attempt to study the robustness of the off-line phase of the system a synthetic image shown in Figure 5.2 and the corresponding ground truth were created. By spatially dilating/eroding the set of ground-truth pixels, labeling errors were introduced into the expert annotated interpretation of the image. Figures 5.3 and 5.4 demonstrate the effect of changing the ground-truth on the optimal interpretation selected by the off-line component of the system.

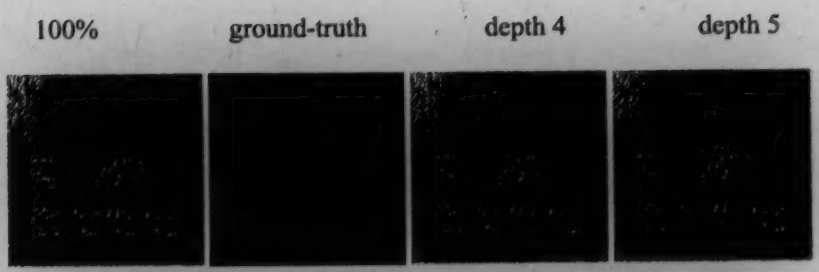


Figure 5.2: Correctly labeled training signal (column 2) to the corresponding input image (column 1). Best interpretation found by MR ADORE during off-line expansion (columns 3 and 4).

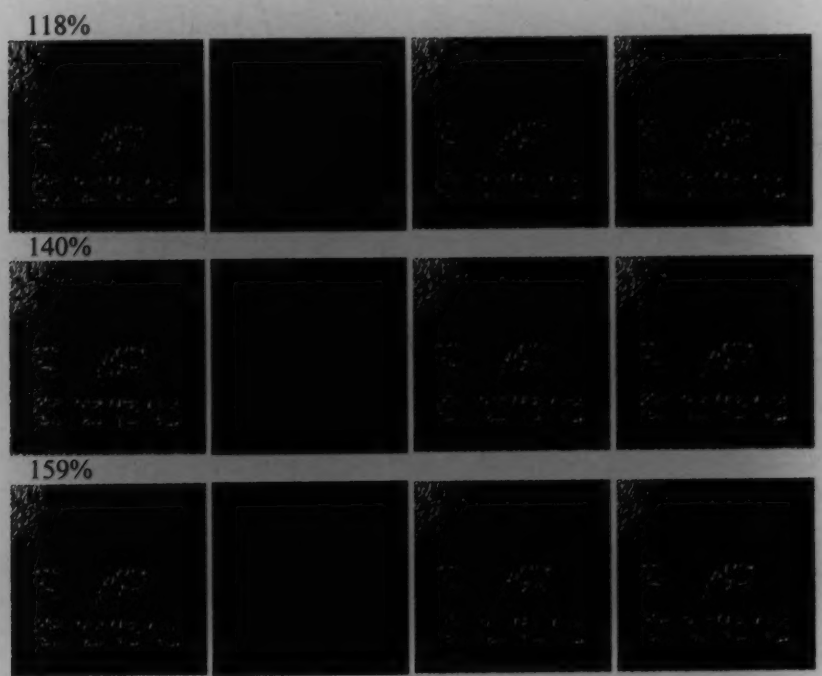


Figure 5.3: Adding pixels to the ground-truth visually has little impact on the off-line optimal image produced. The percentages indicate the number of pixels relative to the true image interpretation depicted in Figure 5.2.

In order to assess the deterioration in performance as a result of labeling errors each interpretation (columns 3 and 4 in Figures 5.3 and 5.4) was re-evaluated against the true hypothesis in Figure 5.2 (second column). The results are presented in Tables 5.6, 5.7, and 5.8.

The initial results indicate that adding pixels (i.e, false positives) to the ground-

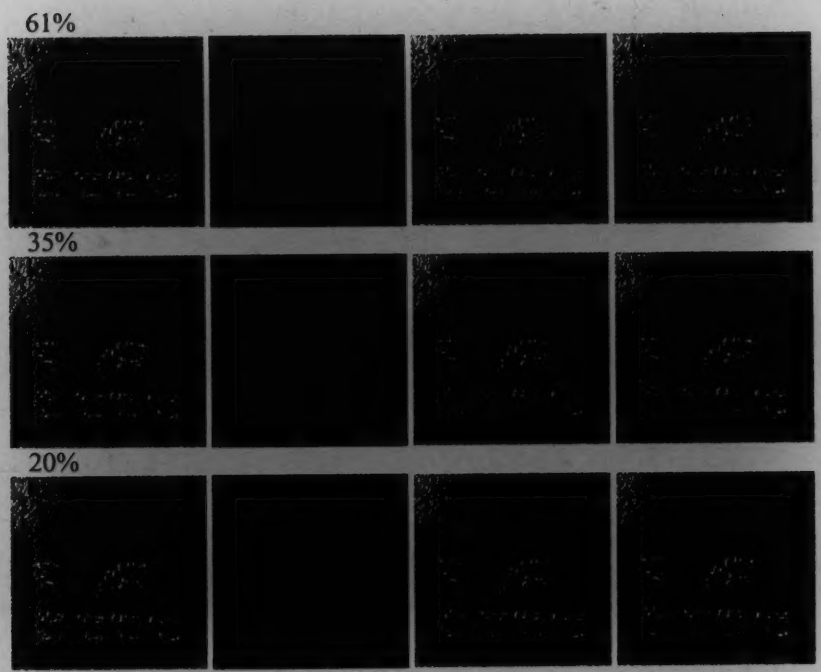


Figure 5.4: Removing pixels from the ground-truth visually *can* have an impact on the off-line optimal image produced. The percentages indicate the number of pixels relative to the true image interpretation depicted in Figure 5.2.

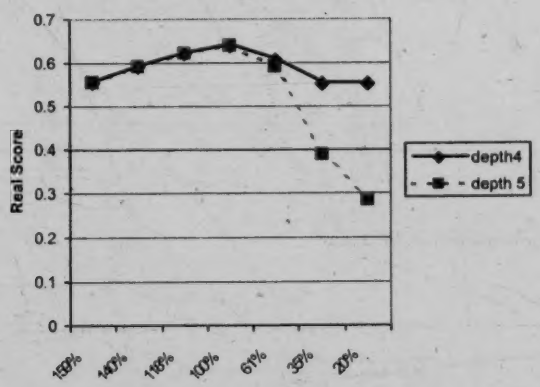
Table 5.6: Off-line robustness to noise. When maximal sequence length is set to four, the experimental results indicate a negligible performance degradation when false positives or false negatives are added to the ground-truth image. Middle row (highlighted) represents the base line performance given the correct ground-truth.

Pixels	%	False Reward	Real Reward
12000	159%	0.77	0.554
10557	140%	0.74	0.592
8915	118%	0.703	0.622
7538	100%	0.639	0.639
4635	61%	0.455	0.606
2626	35%	0.267	0.553
1510	20%	0.147	0.553

Table 5.7: Off-line robustness to noise. When maximal sequence length is set to five (and six), false positives appear less detrimental to the performance of the system when compared to false negatives.

Pixels	%	False Reward	Real Reward
12000	159%	0.77	0.554
10557	140%	0.74	0.592
8915	118%	0.703	0.622
7538	100%	0.64	0.64
4635	61%	0.458	0.592
2626	35%	0.308	0.389
1510	20%	0.2	0.284

Table 5.8: Robustness results. Erroneously adding pixels to the set of target concept pixels appears to be less detrimental than omitting pixels belonging to the target concept for longer sequence lengths.



truth interpretation is less detrimental to performance than missing target class pixels (i.e. false negatives). Table 5.8 shows the performance results of optimal results produced with respect to the erroneous interpretation being compared to the real ground-truth image. Interestingly, depth 6 results are identical to those of depth 5 (not shown), implying that the decrease in the interpretation quality is bounded with respect to the amount of degradation within the ground truth image.

5.5.2 Sun Angle Experiment

Inspired by the interesting results of the previous experiment, another experiment to test the robustness of the system was designed. Motivated by the work in [Culvenor et al., 1999], the experiment varied the positions of the sun while keeping the scene and the corresponding ground-truth constant. Figures 5.5 and 5.6 present the images used in the experiment.

Table 5.9: Off-line performance on the Sun Angle experiment. The current library of operators appears to perform poorly in mid-day sun as the results indicate. Performance analysis has not revealed the reason for this particular phenomenon but it is hypothesized to be a peculiarity of the synthetically generated images.

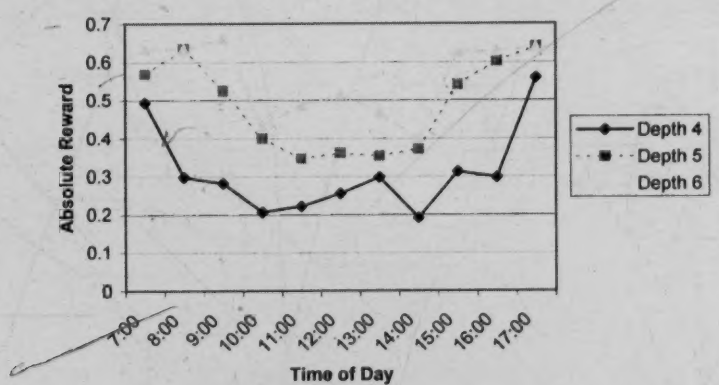


Table 5.9 shows the off-line performance on the sun angle experiment. While the performance appears to degrade during the midday sun, it is the author's opinion that the reflective properties of the synthetic ground covering is to blame. In conjunction with the sparse leaf cover on the target objects, the highly reflective ground is visible through the canopy thereby appearing to be not part of the target objects. Using the SNoW algorithm in conjunction with the texture features, t he on-line policy achieved an 89% accuracy with respect to off-line optimal during the leave-one-out cross-validation experiment.

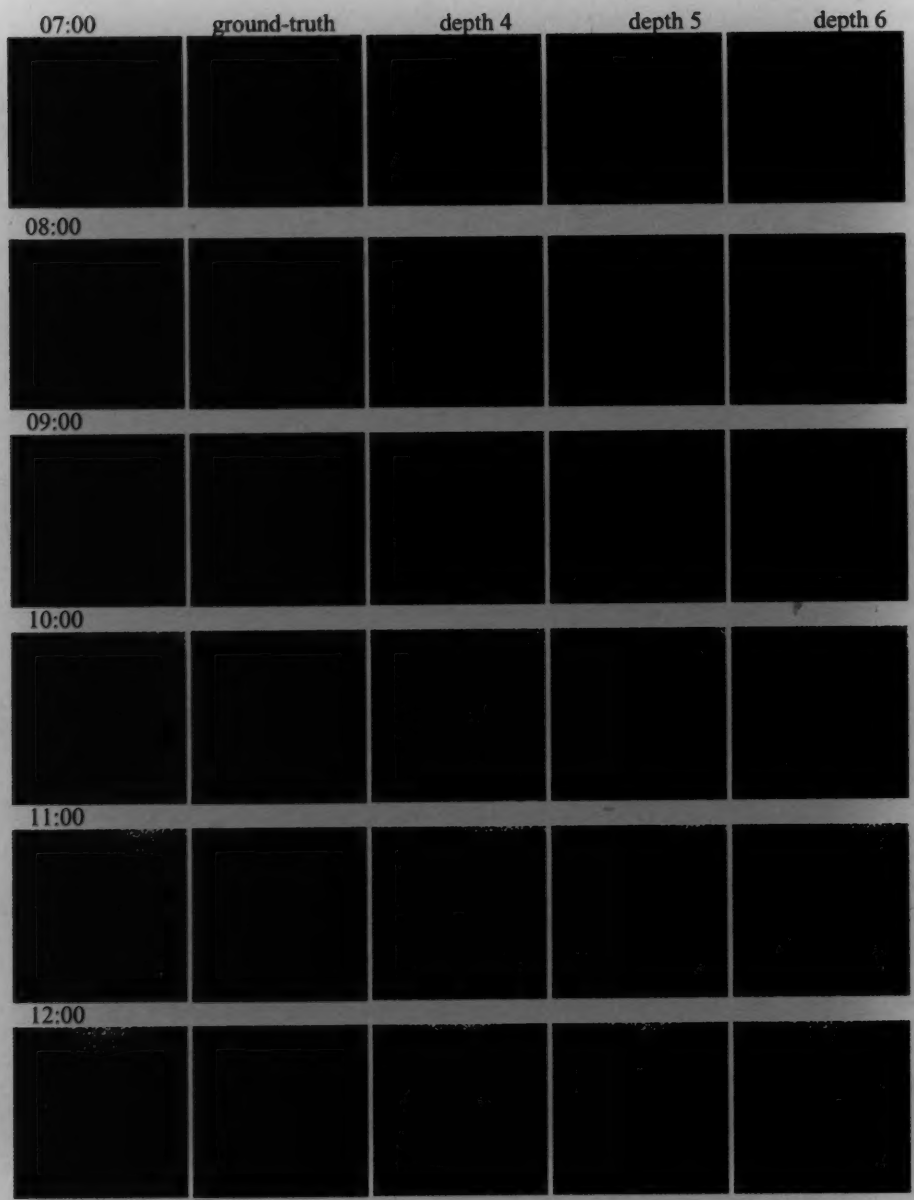


Figure 5.5: Various angles of the sun can produce vastly different image interpretation for a given scene. Sun positions between 07:00 and 12:00 are shown in column one. A static, expert labelled ground-truth image (column 2) is used to compare the hypothesis produced by MR ADORE at various maximal search depths (columns 3-5).

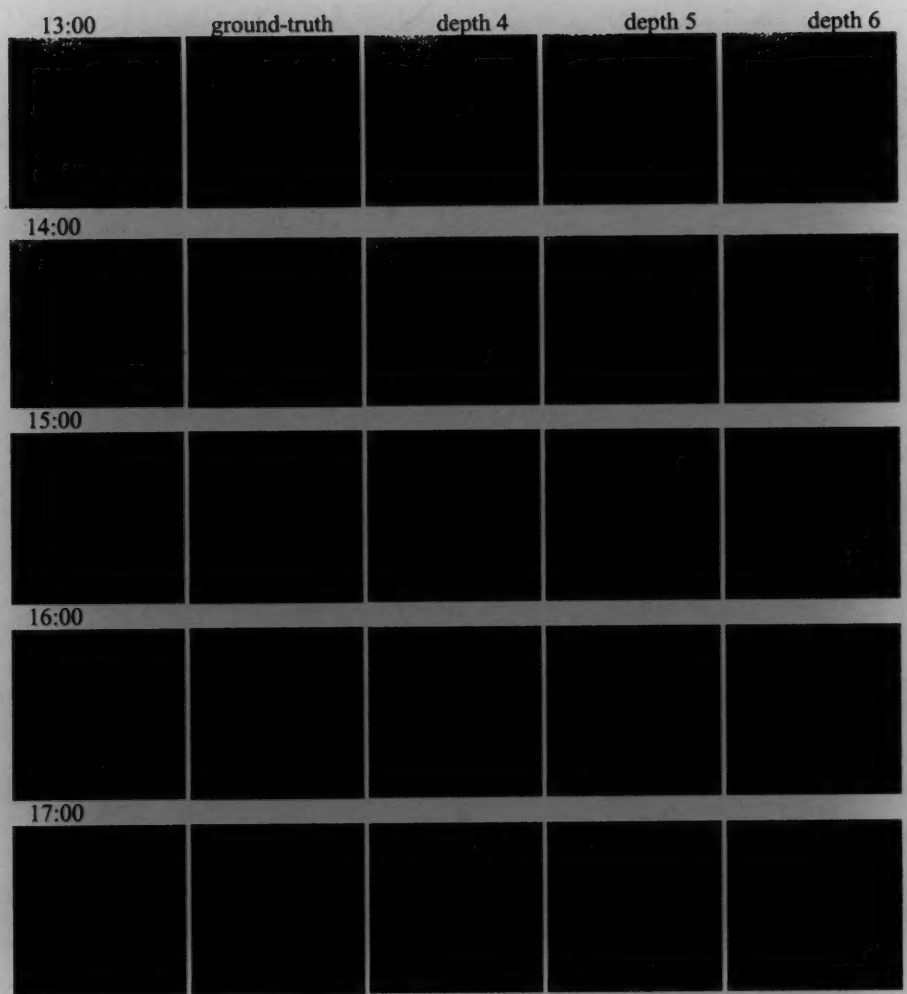


Figure 5.6: Part II of the Sun Angle Experiment. Variation in the angle of the sun results in significantly different image interpretation for a given scene. Sun positions between 13:00 and 17:00 are shown in column one. A static, expert labelled ground-truth image (column 2) is used to compare the hypothesis produced by MR ADORE at various maximal search depths (columns 3-5).

Chapter 6

Conclusions

6.1 Future Research

Although this work presented MR ADORE successfully applied to the domain of forestry, several drawbacks still exist. A problem still exhibited by the framework is the reliance on hand-crafted features. An attempt was made to eliminate features by using PCA in order to compress the images. We used the projection of an image onto a PCA basis as a standard feature vector used as input by the reward approximator of state-action pairs. The technique has not performed as well as other novel techniques presented in this thesis. However, the proposed approach aimed at reducing the need for features by eliminating control points at all but the lowest data layer within MR ADORE was successful. The least-commitment policy needs only a single set of hand-crafted features and also requires less off-line training time. As a consequence less resources can be spent for design and development of machine-learned control points. As a drawback, the system is now forced to perform *full search-tree expansion* (to a maximal depth) resulting in increased online run-time[1].

Although the on-line runtime increases, the production of all possible hypothesis tokens can actually be a useful consequence of this approach. The next section explores a possible future research direction utilizing multiple hypotheses in order to improve the overall performance of the system.

6.1.1 Hypothesis merging

We introduce the concept of hypotheses merging by reproducing a quote (already stated in chapter 2) by Axel Pinz:

"No single algorithm is able to solve the complete task of tree finding and species classification. A hybrid multi-level approach to scene interpretation, where several competing hypothesis are established and evaluated is required" [Pinz, 1998b].

Currently MR ADORE produces many hypotheses, which are ranked according a machine learned version of $\frac{A \cap B}{A \cup B}$. However, some operator sequences tend to perform well on specific subparts of an image while different operators perform well on different areas of the image. Illustrated in Figure 6.1, the system can take advantages of such performance and *merge* the output of individual sequences into a hypothesis that surpasses in quality any of the initial hypotheses.

[1]The old ADORE had the ability to backtrack and could therefore also perform a full tree expansion. In addition since control points were used at every possible decision point, the run time can actually be greater than that of MR ADORE. According to [Draper, 2002], the system would either submit the result of the first token, produced online, or would perform a full search up to the maximal allowed depth.

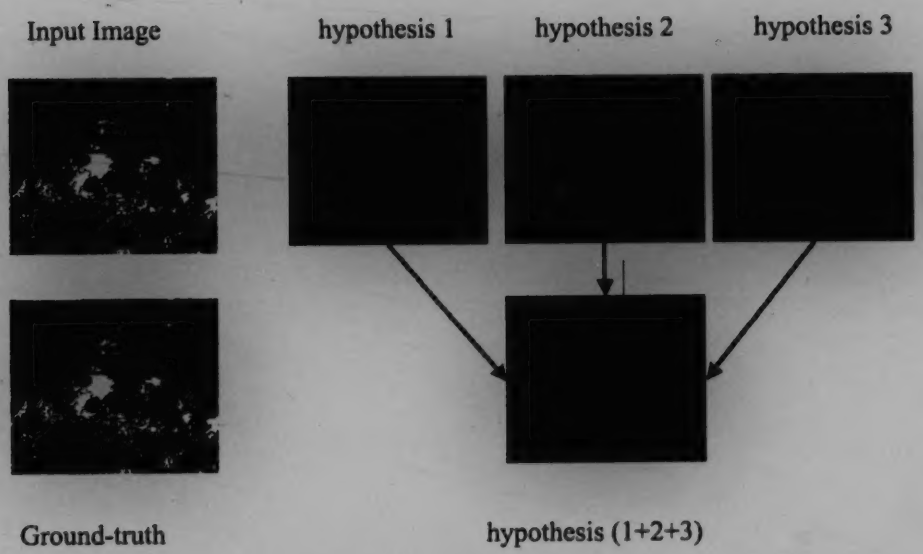


Figure 6.1: An envisioned scenario of hypothesis merging. By merging together hypotheses 1, 2, and 3 a new hypothesis (1+2+3) is a much better interpretation of the image than any singe hypothesis.

6.1.2 Incremental Focus of Attention

An alternative means to taking advantage of sequences specializing on various image subparts is to simply split the initial image into sub-images. Commonly known as "Focus of Attention" or FOA algorithms, such sub-systems select regions of interest for further processing. By *learning* how to split an image into, possibly overlapping sub-images, the performance of the system can be significantly improved. However, after all sub-images have been interpreted, the system needs to regroup them into one coherent interpretation of the whole image. Therefore, it seems hypotheses merging in again needed to improve the performance of the system. Figure 6.2 presents a possible future object recognition system.

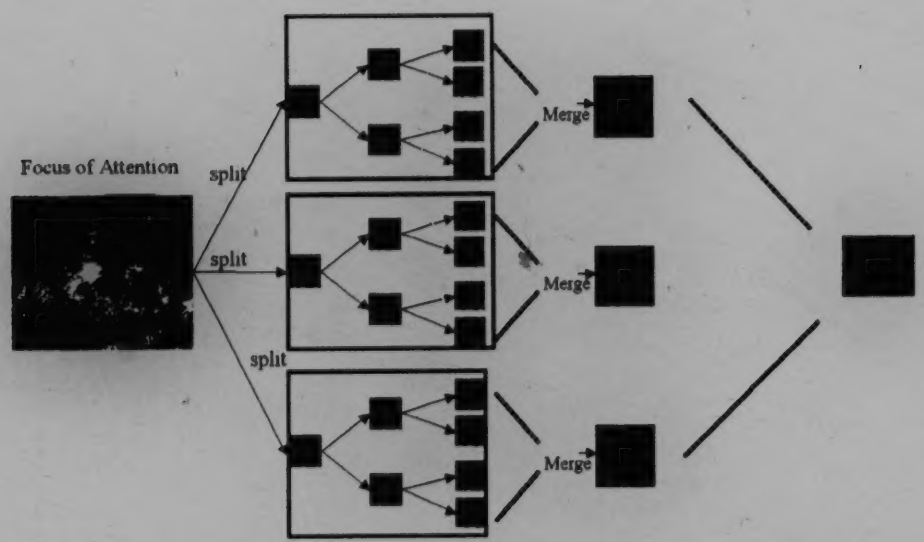


Figure 6.2: A vision of the future object recognition systems. Focus of attention *adaptively* selects image regions to be processed by MR ADORE. The individual hypotheses are merged to form a single interpretation of the sub-image. All sub-image interpretation are then merged again into a single coherent interpretation of the whole image.

6.2 Conclusion

Object recognition, also known as image interpretation, has numerous practical applications in the modern society. To date, image interpretation systems can only operate in highly controlled domains under close human supervision. Such systems are difficult to engineer, maintain, and port to other domains. Spurred by advances in machine learning technology and increased computational resources, recently researchers have turned to exploring more automated approaches for developing object recognition systems. The aim of this thesis was to investigate one of the main barriers to automatic creation of vision systems, namely the need for hand-engineered features. While the ideal solution to the problem still remains undiscovered, the research presented in this thesis has demonstrated several approaches for increasing the level of automation in image interpretation systems. In the opinion of the author, human intervention will probably not be eliminated by a single approach. Rather, removing the last vestiges of human intervention will require unique solutions tailored to the task at hand. Thus, solutions to automated feature extraction problem may need to use a library of feature extraction algorithms, requiring the system to perform trail-and-error experimentation in order to *learn* which features enable the system to solve the given image interpretation task.

Bibliography

Andress, K., & Kak, A. (1988). Evidence accumulation and flow of control in a hierarchical spatial reasoning system. *AI Magazine, 9*, 75–94.

Arbib, M. (1978). Segmentation, schemas, and cooperative computation. In S. Levin (Ed.), *MAA studies in Mathematics*, 118–155.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM, 45*, 891–923.

Au, W., & Roberts, B. (1996). Adaptive configuration and control in an ATR system. *Proceedings of the DARPA Image Understanding Workshop* (pp. 667–676). Palm Springs, CA.

Avery, T., & Berlin, G. (1992). *Fundamentals of remote sensing and airphoto interpretation*. New York, NY: Macmillan Publishing Company. 5 edition.

Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence, 72*, 81–138.

Bhanu, B., Lee, S., & Ming, J. (1995). Adaptive image segmentation using a genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetocs, 25*, 1543–1568.

Bhanu, B., & Peng, J. (2000). Adaptive integrated image segmentation and object recognition. *IEEE Transactions on Systems, Man, and Cybernetocs - Part C: Application And Reviews, 30*, 427–441.

Blazquez, C. (1989). Computer-based image analysis and tree counting with aerial color infrared photography. *Journal of Imaging Technology, 15*.

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Artificial Intelligence Research, 11*.

Brandtberg, T. (1999). Remote sensing for forestry applications - a historical retrospect. *http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/BRANDTBERG/UK.html*.

Bulitko, V., & Levner, I. (2003). *Improving learnability of adaptive image interpretation systems* (Technical Report). University of Alberta.

Burl, M. C., Asker, L., Smyth, P., Fayyad, U. M., Perona, P., Crumpler, L., & Aubele, J. (1998). Learning to recognize volcanoes on venus. *Machine Learning, 30*, 165–194.

Chien, S., Mortensen, H., Ying, C., & Hsiao, S. (1995). Integrated planning for automated image processing. *Integrated Planning Applications, AAAI Spring Symposium Series*, 26–35.

Clement, V., & Thonnat, M. (1993). A knowledge-based approach to integration of image processing procedures. *CGVIP, 57*, 166–184.

Cormen, T., Leiserson, C., & Rivest, R. (1990). *Introduction to algorithms*. New York: MIT Electrical Engineering and Computer Science Series. MIT Press/McGraw-Hill.

Culvenor, D., Prestor, R., & Tolhurst, K. (1999). A spacial clustering approach to automated tree crown delineation. *In Proceedings of the International Forum on Automated Interpretation of High Spacial Resolution Digital Imagery for Forestry* (pp. 67–80).

Daley, N., Burnett, C., Wulder, M., Niemann, K., & Goodenough, D. (2000). Error reduction methods for local maximum filtering. *The Proceedings of the 22nd Symposium of the Canadian Remote Sensing Society*. Victoria, British Columbia.

Draper, B. (1993). *Learning object recognition strategies*. Doctoral dissertation, University of Massachusetts, Depertment of Computer Science.

Draper, B. (1996a). Modelling object recognition as a markov decision process. *International Conference on Pattern Recognition*, D95–99.

Draper, B. (1996b). Object recognition as a markov decision process. *Proceedings of the International Conference on Pattern Recognition* (pp. 95–99). Vienna, Austria.

Draper, B. (1997). Learning control strategies for object recognition. In K. Ikeuchi and M. Veloso (Eds.), *Symbolic visual learning*, 49–76. New York: Oxford University Press.

Draper, B. (2002). Personal communication.

Draper, B., Ahlrichs, U., & Paulus, D. (2001). Adapting object recognition across domains: A demonstration. *Proceedings of International Conference on Vision Systems* (pp. 256–267). Vancouver, B.C.

Draper, B., Bins, J., & Baek, K. (2000). ADORE: adaptive object recognition. *Videre, 1*, 86–99.

Draper, B., Collins, R., Brolio, J., Hanson, A., & Riseman, E. (1989). The schema system. *International Journal of Computer Vision, 2*, 209–250.

Draper, B., & Hanson, A. (1992). An example of learning in knowledge directed vision. In *Theory and applications of image analysis*, 237–252. Singapore: World Scientific.

Draper, B., Hanson, A., & Riseman, E. (1996). Knowledge-directed vision: Control, learning and integration. *Proceedings of the IEEE, 84*, 1625–1637.

Draper, B. A. (2003). From knowledge bases to Markov models to PCA. *Proceedings of Workshop on Computer Vision System Control Architectures*. Graz, Austria.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: John Wiley & Sons.

Goldberg, D. (1987). *Genetic algorithms in search, optimization, and machine learning*. Addison-Westley.

Gonzalez, R. C., & E.Woods, R. (2002). *Digital image processing*. Prentice Hall. 2 edition.

Gougeon, F. (1993). Individual tree identification from high resolution meis images. *Proc. Int'l Forum on Airborne Multispectral Scanning for Forestry and Mapping* (pp. 117–128). Forestry Canada, Chalk River, Ontario.

Gougeon, F. (1995a). Comparison of multispectral classification schemes for tree crowns individually delineated on high spatial resolution meis images. *Canadian Journal of Remote Sensing, 21*, 1–9.

Gougeon, F. (1995b). A system for individual tree crown classification of conifer stands at high spatial resolutions. *Proceedings of the 17th Canadian Symposium on Remote Sensing* (pp. 635–642).

Gougeon, F. (1997a). A locally adaptive technique for forest regeneration assessments from high resolution aerial images. *Proceedings of the 19th Canadian Remote Sensing Symposium.*

Gougeon, F. (1997b). Recognizing the forest from the trees: Individual tree crown deleneation, classification and regrouping for inventory purposes. *Proceedings of the Third International Airborne Remote Sensing Conference and Exhibition.*

Gougeon, F. (1998). Automatic individual tree crown deleniation using a valley-following approach and a rule-based system. *Automated Interpretation of High Spacial Resolution Digital Imagery for Forestry.*.

Gougeon, F., & Leckie, D. (2001). Individual tree crown image analysis - a step towards precision forestry. *First Int. Precision Forestry Symposium.*

Gougeon, F., & Leckie, D. (2003). *Forest information extraction from high spatial resolution images using an individual tree crown approach* (Technical Report). Pacific Forestry Centre.

Gougeon, F., & Moore, T. (1988). Individual tree classification using meis-ii imagery. *Proceedings Geoscience and Remore Sensing Symposium* (pp. 13–16).

Greiffenhagen, M., Comaniciu, D., Niemann, H., & Ramesh, V. (2001). Design, analysis, and engineering of video monitoring systems: An approach and a case study. *Proceedings of the IEEE, 89*, 1498–1517.

Hall, P., Martin, R., & Marshall, A. (2000). Merging and splitting eigenspace models. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 22, 1042–1050.

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. Macmillian College Pub. Co.

Hwang, V., Davis, L., & Matsuyama, T. (1986). Hypothesis integration in image understanding systems. *CGVIP*, *36*, 321–371.

Isukapalli, R., & Greiner, R. (2003). Use of off-line dynamic programing for efficient image interpretation. *Proceedings of the International Joint Conference on Artificial Intelligence*.

Jiang, X., & Bunke, H. (1994). Vision planner for an intelligent multisensory vision system. *Automatic Object Recognition IV*, 226–237.

Kirby, M. (2001). *Geometric data analysis: An emprical approach to dimensionality reduction and the study of patterns*. New York: John Wiley & Sons.

Korf, R. E. (1990). Real-time heuristic search. *Artificial Intelligence*, *42*, 189–211.

Korpela, I. S. (2002). *3-d matching of tree tops using digitized panchromatic aerial photos*. Licentiate in agriculture and forestry, University of Helsinki.

Lansky, A., Friedman, M., Getor, L., Schmidler, S., & Short, N. J. (1995). The COLLAGE/KHOROS link: Planning for image processing tasks. *Integrated Planning Applications, AAAI Spring Symposium Series*, 67–76.

Larsen, M. (1997). Crown modeling to find tree top positions in aerial photographs. *Proceedings of the Third International Airborne Remote Sensing Conference and Exhibition*.

Larsen, M. (1999). Individual tree top position estimation by template voting. *Proceedings of the Fourth International Airborne Remote Sensing Conference and Exhibition*.

Larsen, M., & Rudemo, M. (1997). Estimation of tree positions from aerial photos. *Proceedings of the*.

Leckie, D., & Gougeon, F. (1998). An assessment of both visual and automated tree counting and species identification with high spatial resolution multispectral imagery. *Automated Interpretation of High Spacial Resolution Digital Imagery for Forestry*. (pp. 141–152).

Maloof, M., Langley, P., Sage, S., & Binford, T. (1997). Learning to detect rooftops in aerial images. *Proceedings of the Image Understanding Workshop* (pp. 835–845). San Francisco, CA: Morgan Kaufmann.

Mann, W., & Binford, T. (1996). Successor: Interpretation overview and constraint system. *IUW*, 1505–1518.

McKeown, D., Harvey, W., & McDermott, J. (1985). Rule-based interpretation of aerial imagery. *PAMI*, *7*, 570–585.

Murgu, D. (1996). Individual tree detection and localization in aerial imagery. Master's thesis, Department of Computer Science, University of British Columbia.

Nagao, M., & Matsuyama, T. (1980). *A structural analysis of complex aerial photographs*. N.Y.: Plenum Press.

Ojala, T., & Pietikinen, M. (1999). Unsupervised texture segmentation using feature distributions. *Pattern Recognition, 32*, 477–486.

Pelleg, D., & Moore, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases* (pp. 277–281). AAAI Press.

Peng, J., & Bhanu, B. (1998). Closed-loop object recognition using reinforcement learning. *PAMI, 20*, 139–154.

Petersen, P. (2001). Finding single trees in aerial imagery. Masters thesis, Royal Veterinary and Agricultural University of Copenhagen.

Pinz, A. (1991). A computer vision system for the recognition of trees in aerial photographs. *J.C.Tilton (ed.), Multisource Data Integration in Remote Sensing* (pp. 111–124). Maryland: NASA.

Pinz, A. (1998a). Austrian forest inventory system. *International Forum on Automated Interpretation of High Spacial Resolution Digital Imagery for Forestry.*.

Pinz, A. (1998b). Tree isolation and species classification. *International Forum on Automated Interpretation of High Spacial Resolution Digital Imagery for Forestry.*.

Pollock, R. (1994). A model-based approach to automatically locating tree crowns in high spatial resolution images. *Image and Signal Processing for Remote Sensing*.

Pollock, R. (1996). *The automatic recognition of individual trees in aerial images of forests based on a synthetic tree crown image model*. Doctoral dissertation, University of British Columbia, Vancouver, Canada.

Pouliot, D., King, D., Bell, F., & Pitt, D. (2002). Automated tree crown detection and delineation in high-resolution digital camera imagery of coniferous forest regeneration. *Remote Sensing of Environment, 82*.

Rimey, R., & Brown, C. (1994). Control of selective perception using bayes nets and decision theory. *International Journal of Computer Vision, 12*, 173–207.

Rudemo, M. (1998). Spacial tree patterns analysis from maxima of smoothed aerial photographs. *Automated Interpretation of High Spacial Resolution Digital Imagery for Forestry.*.

Sirovich, L., & Kirby, M. (1987). Low dimensional procedure for the characterization of human faces. *Journal of Optical Society of America, 4*, 519–524.

Sutton, R., & Barto, A. (2000). *Reinforcement learning: An introduction*. MIT Press.

Thompson, M. (1966). *Manual of photogrammetry*. Falls Church, VA: American Society of Photogrammetry. 3 edition.

Toumey, J., & Korstian, C. F. (1947). *Foundations of silviculture upon an ecological basis*. New York, NY: John Wiley & Sons. 3 edition.

Wang, L., Gong, P., & Biging, G. (2002). Automated individual tree crown delineation and treetop detection in high-spatial resolution aerial imagery. *UCGIS*.

Wulder, M., Niemann, K., & Goodenough, D. (2000a). Error reduction methods for local maximum filtering. *The Proceedings of the 22nd Symposium of the Canadian Remote Sensing Society*. Victoria, British Columbia.

Wulder, M., Niemann, K., & Goodenough, D. (2000b). Local maximum filtering for the extraction of tree locations and basal area from high spatial resolution imagery. *Remote Sensing of Environment, 73*, 103–114.

Appendix A

Vision Operators within MR ADORE

A.1 Vision Operators

All vision operators are located in `/MrAdore/NewKisr/applications`
Figure 4.2 displays the data layers currently present within MR ADORE.

A.1.1 Basic Operations

These miscellaneous operations consist of Grabbing an Image, converting an image from Color to Grayscale and also from Grayscale to Black and White (binary) image.

Grab Image

```
Usage: /bin/cp input output
```

Convert Image

```
Usage: /usr/bin/X11/convert input.pnm output.pnm
    Convert uses the filename extensions to infer the type
    conversion requested. (I.e., convert a.ppm a.pbm will convert a
    color image to a binary image).
```

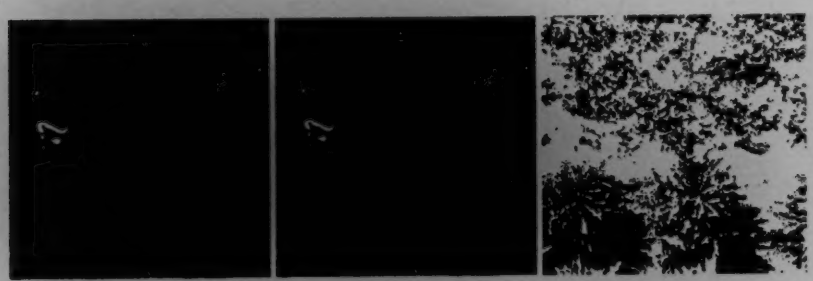


Figure A.1: Left to Right: Original color image. Grayscale and binary images produced by Unix convert function.

A.1.2 Image Filters

For more information on Spacial Image Filtering see Chapter 3 of [Gonzalez & E.Woods, 2002].

```
Usage: filter infile.pnm outfile.pnm g|l|h window_size
The following filters support window sizes of 3 or 5:
    Filter: g (Gaussian)
    Filter: l (LaPlacian)
    Filter: h (HiPass)
The following filters support any (> 1) size window:
    Filter: m (Median)
    Filter: n (Min)
    Filter: x (Max)
```

Gaussian Filter

A lowpass filter. (i.e., A filter which removes high frequency components) Visually, the gaussian filter, has the effect of smoothing out the image. The filter works by replacing a pixel with its neighborhood average. In spacial domain, the typical 3×3 masks are:

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \; ; \; \frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \tag{A.1}$$

The larger the convolution mask the larger the extent of the smoothing will be. (I.e., larger and larger masks remove lower and lower frequency components as seen in the following example).

Figure A.2: Left to Right: Original Image. An image filtered with a 3×3 Gaussian. An image filtered with a 5×5 Gaussian.

Currently a 5×5 Gaussian filter is used in MR ADORE.

Median

Median filter, belongs to the *order-statistics* family of filters, and replaces a pixels value by the median intensity of neighboring pixels. The median is a value such that half the values in the neighborhood are less then median and the other half is greater than the median value. This filter is effective for removing *salt and pepper* noise. For a 3×3 neighborhood the 5^{th} largest value is the median and the 13^{th} largest value is the median for a 5×5 mask. Currently a 5×5 Median filter is used in MR ADORE.

Min

This filter replaces the value of a pixel by the lowest (0^{th} percentile) intensity value within a given neighborhood. A min filter amplifies the darkest points within an

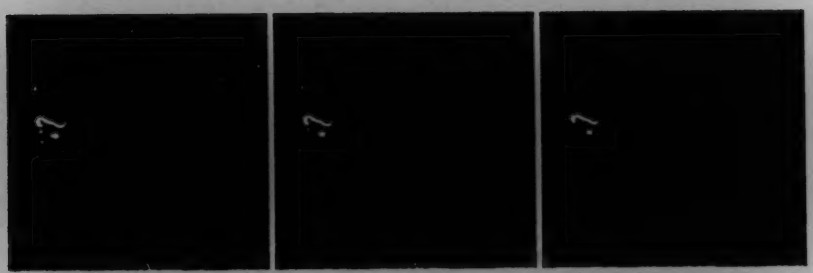


Figure A.3: Left to Right: Original Image. An image filtered with a 3×3 Median filter. An image filtered with a 5×5 Median filter.

image.

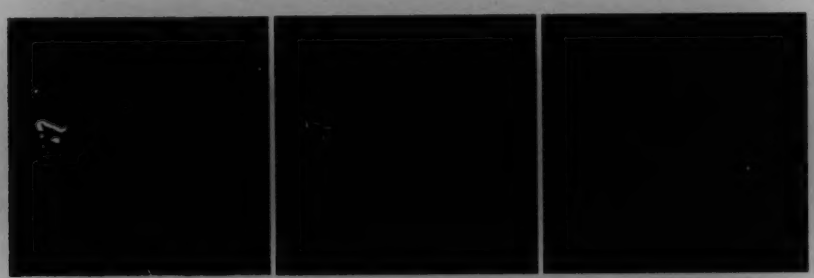


Figure A.4: Left to Right: Original Image. An image filtered with a 3×3 Min filter. An image filtered with a 5×5 Min filter.

Currently a 5×5 Min filter is used in MR ADORE.

Max

This filter replaces the value of a pixel by the highest (100^{th} percentile) intensity value within a given neighborhood. A max filter amplifies the brightest points within an image.
Currently a 5×5 Max filter is used in MR ADORE.

Laplacian

In contrast to the previous filters, the Laplacian sharpens the image. The filter uses second order discrete intensity derivatives to highlight edges (discontinuities in intensity) within an image. Typical convolution masks are:

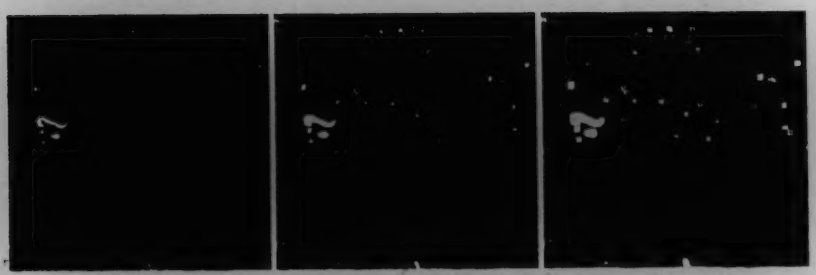


Figure A.5: Left to Right: Original Image. An image filtered with a 3×3 Max filter. An image filtered with a 5×5 Max filter.

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & -4 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \; ; \; \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & -8 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \tag{A.2}$$

Figure A.6: Left to Right: Original Image. An image filtered with a 3×3 Laplacian filter. An image filtered with a 5×5 Laplacian filter.

Currently the Laplacian is not used within MR ADORE. However, the desired effect is to add the result of the Laplacian filter, back to the original to produce Figure A.7.

HiPass

As the name implies the this filter passes high frequency components while blocking low frequency components and is similar in effect to the Laplacian filter.

Currently the HiPass is not used within MR ADORE for the same reasons as the Laplacian. What we want is to add the result of the HiPass filter, back to the original image which would produce A.9.

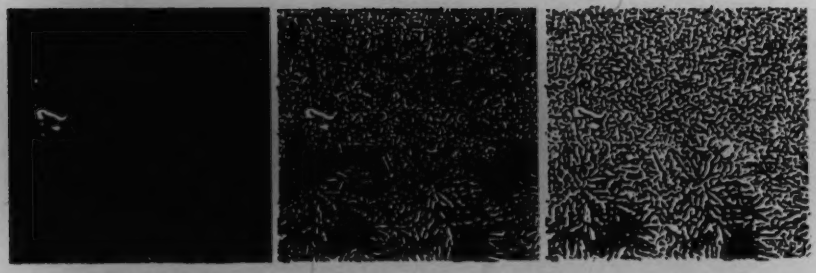


Figure A.7: Left to Right: Original Image. An image filtered with a 3×3 Laplacian filter and added to original image. An image filtered with a 5×5 Laplacian filter and added to original image.

Figure A.8: Left to Right: Original Image. An image filtered with a 3×3 HiPass filter. An image filtered with a 5×5 HiPass filter.

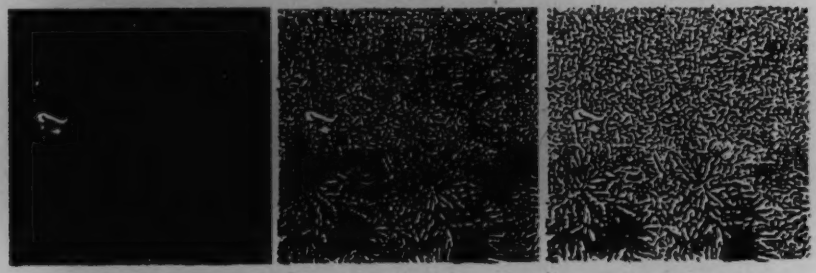


Figure A.9: Left to Right: Original Image. An image filtered with a 3×3 HiPass filter and added to original image. An image filtered with a 5×5 HiPass filter and added to original image.

A.1.3 Morphological Image Processing

For more information on Mathematical Morphology see Chapter 9 of[Gonzalez & E.Woods, 2002],

```
Usage: morphMulti infile.pgm ListFile fileType MorphType KernelType iteration
MorphType:  erode-(e), dilate-(d), close-(c), open-(o), deltaA(g)
KernelType: structuring element used in operation (currenly only 3x3 with zero ofsets)
            rectange(r), cross(c), ellipse(e)
Iterations: the maximum number of times a morph operation is to be applied
NOTE: The function produces a list of created images (filetype) with for 1 to number of iterations
```

The common convolution kernels, shown bellow are the rectangle, cross and ellipse (respectively):

$$\begin{array}{|c|c|c|}\hline 1&1&1\\\hline 1&0&1\\\hline 1&1&1\\\hline\end{array}\quad;\quad\begin{array}{|c|c|c|}\hline 0&1&0\\\hline 1&1&1\\\hline 0&1&0\\\hline\end{array}\quad;\quad\begin{array}{|c|c|c|}\hline 0&1&0\\\hline 1&0&1\\\hline 0&1&0\\\hline\end{array} \tag{A.3}$$

Currently only the ellipse is used (chosen arbitrarily, since the small 3×3 kernels all behave similarly).

Erode

Denoted as $A \ominus B$, the morphological erode operation is defined as

$$(A \ominus B)(s,t) = min\{A(s+x,t+y) - B(x,y) \mid (s+x,t+y) \in D_A; (x,y) \in D_B\} \tag{A.4}$$

where D_A and D_B are the domains of sets A and B. Erosion has a general property of darkening the image and visually produces results similar to a Min filtering operation. Multiple application of Erode produce darker and darker images similar to using a larger and larger convolution mask for a Min filter.

Figure A.10: Left to Right: Original Image. An image morphed by an Erode operation with: a rectangular mask, a cross mask, and an elliptical mask.

Dilate

Denoted as $A \oplus B$, the morphological dilate operation is defined as:

$$(A \oplus B)(s,t) = max\{A(s-x,t-y)+B(x,y) \mid (s+x,t+y) \in D_A; (x,y) \in D_B\} \quad \text{(A.5)}$$

where D_A and D_B are the domains of sets A and B. Dilate has a general property of brightening the image and visually produces results similar to a Max filtering operation. Multiple applications of Dilate produce darker and darker images similar to using a larger and larger convolution mask for a Max filter.

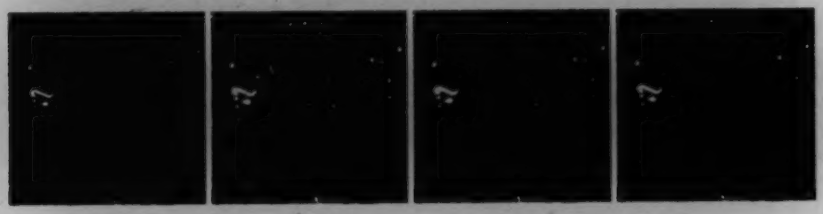


Figure A.11: Left to Right: Original Image. An image morphed by a Dilate operation with: a rectangular mask, a cross mask, and an elliptical mask.

Opening

Defined as $A \circ B = (A \ominus B) \oplus B$, (I.e., an erosion followed by dilation). Morphological Opening removes small (w.r.t convolution kernel) bright details.

Figure A.12: Left to Right: Original Image. An image morphed by an Open operation with: a rectangular mask, a cross mask, and an elliptical mask.

Closing

Defined as $A \bullet B = (A \oplus B) \ominus B$, (I.e., a dilation followed by erosion). Morphological Closing removes the small (w.r.t convolution kernel) dark details leaving the rest of the image relatively undisturbed. Applying morphological closing to a forestry scene should (in theory) remove the shadows cast by individual leaves.

Figure A.13: Left to Right: Original Image. An image morphed by a Close operation with: a rectangular mask, a cross mask, and an elliptical mask.

Morphological Smoothing

By applying morphological opening followed by closing an image can be smoothed as illustrated in figure A.14.

Figure A.14: Left to Right: Original Image. An image smoothed by applying morphological Opening followed by morphological Closing operation with a cross mask. An image smoothed by applying morphological Closing followed by morphological Opening operation with a cross mask.

These operations are implicitly possible within MR ADORE.

Morphological Gradient

Defined as $g = (A \oplus B) - (A \ominus B)$

This operator is not currently used in MR ADORE.

A.1.4 Cross Match Operators

```
Usage: crossmatch datafile.ppm templatefile.ppm outfile.pgm c|h|i|k|m|r [edge_thresh : r only]
Usage: colorcrossmatch datafile.ppm templatefile.ppm outfile.pgm c|h|i|k|m|r [edge_thresh : r only]
    This program passes a window of size templatefile across each color plane of the datafile,
    comparing the window to templatefile at every position. The color planes are then 'AND'ed
    to produce a grayscale probability map.
The match measures are:
    Match: c (correlation)
    Match: h (histogram/chi square)
```

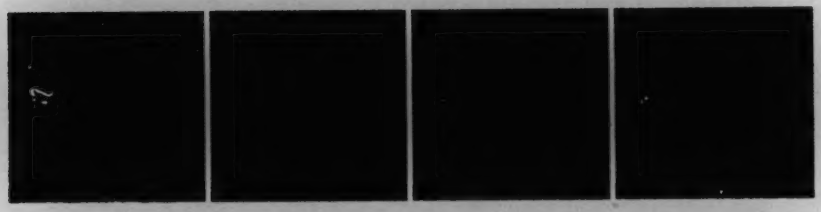


Figure A.15: Left to Right: Original Image. Edges extracted using morphological gradient $g = (A \oplus B)^1 - (A \ominus B)^1$. Edges extracted using morphological gradient $g = (A \oplus B)^2 - (A \ominus B)^2$. Edges extracted using morphological gradient $g = (A \oplus B)^3 - (A \ominus B)^3$.

```
Match: i (histogram/intersection)
Match: k (histogram/Kolmogorov-Smirnov)
Match: m (mutual information)
Match: r (rotational cross correlation)
```

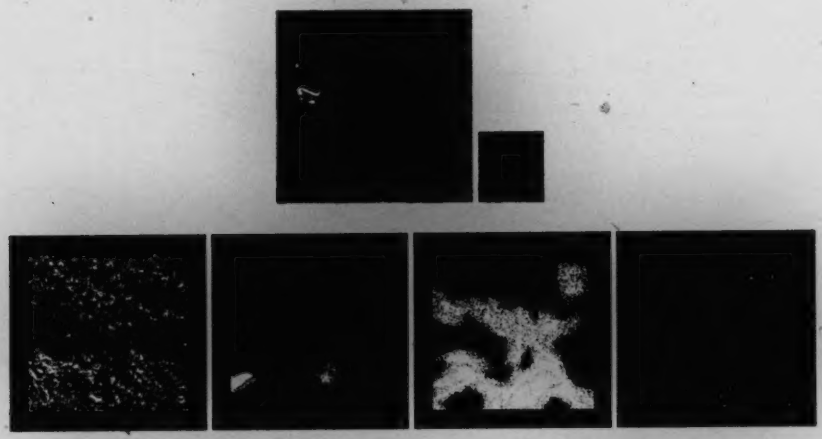


Figure A.16: Top: Original Color Image and Color Template Image. Bottom: Probability maps for: Cross Correlation, Histogram Intersection, Mutual Information, and Rotational Cross Correlation.

A.1.5 Color Segmentation

```
Usage: segmRGB infile.ppm outfile.pbm template.ppm
    Produces a binary mask where RGB color intensities
    are within 1 stDev of template color intensities
```

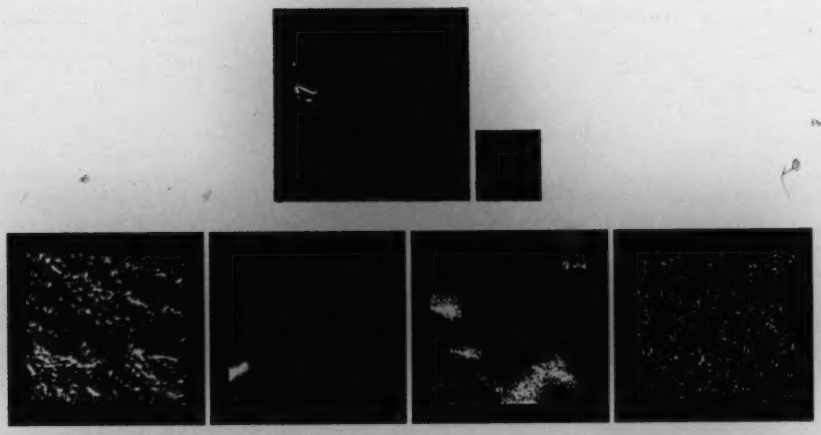


Figure A.17: Top: Original Image and Template Image. Bottom: Probability maps for: Cross Correlation, Histogram Intersection, Mutual Information, and Rotational Cross Correlation.

The operator calculates the average color brightness and standard deviation for each color plane of the template. Then each pixel in the source image is tested to see if it is within one standard deviation of average intensity of the template image. If within the range, the pixel is deemed to belong to a label class.

A.1.6 Pyramid Segmentation

```
Usage: infile.gpm outfile.pgm level treshl thresh2
    level: Maximum level of the pyramid. For now, level must be 3 or less.
    threshl: Error threshold for establishing links.
    thresh2: Error threshold for the segment clustering.
```

A.1.7 Histogram Equalization

```
Usage: histogram src.pnm dest.pnm
    Images can be color or grayscale.
```

A.1.8 Thresholding

A.1.9 Resize

```
Usage: resize a|r src.pnm dest.pnm xsz ysz [min_size]
    Resizes an image by either relative or absolute amounts.
    If mode is ABSOLUTE (a), then the dest image will be xsz by ysz pixels.
    If mode is RELATIVE (r), then the dest image will by src->width*xsz by src->height*ysz.
    [min_size] is an optional 6th parameter. If the dest image size is less than min_size,
    no output is created.
```

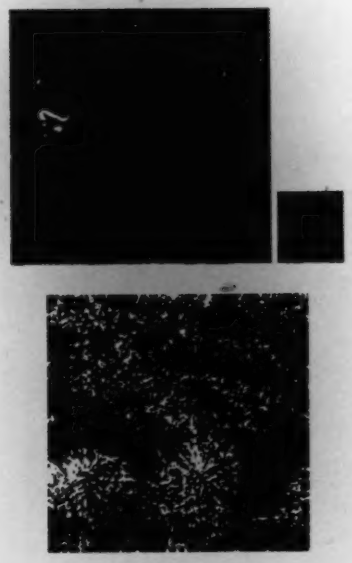


Figure A.18: Top: Original Image and the template. Bottom: The resulting image after Color Segmentation.

Figure A.19: Original Image and the resulting image after Pyramid Segmentation.

A.1.10 Submit

```
Usage: submitImage unlabled_image.ppm labeled_image.ppm submitimage.pbm finalimage.ppm
       rewardfile.txt r g b [Tolerance = 5]

    This program assumes that the full image is labelled, in that the target object has
    a single, known color. This color is given by the (r,g,b) arguments. The program
    writes to rewardfile.txt the percent of the pixels labelled correctly.
    The level of resolution of the region is inferred from its filename,
    and the full_image is reduced to a matching level..
    Tolerance is the amount of color variation tolerated between labelled and submitted images.
```

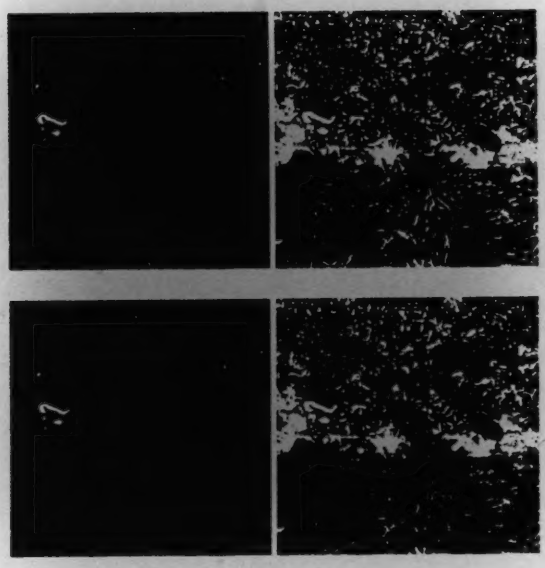


Figure A.20: Top: Original Grayscale Image and resulting image after Histogram Equalization. Bottom: Original Color Image and resulting image after Histogram Equalization.

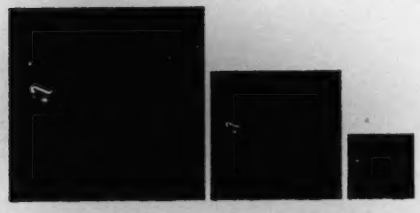


Figure A.21: Left to Right: Original Image. Resulting image after Resizing once and twice.

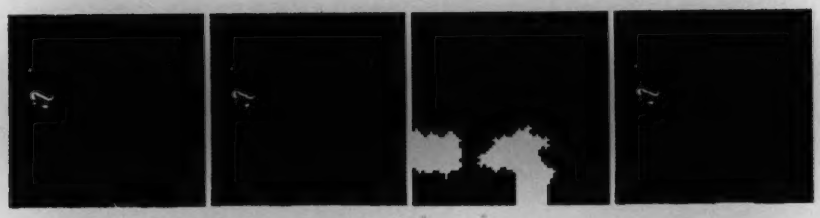


Figure A.22: Left to Right: Unlabelled original image. Labelled original image Segmented image. Final Image. Reward produced is 0.5004053 (out of a maximum of 1.0)

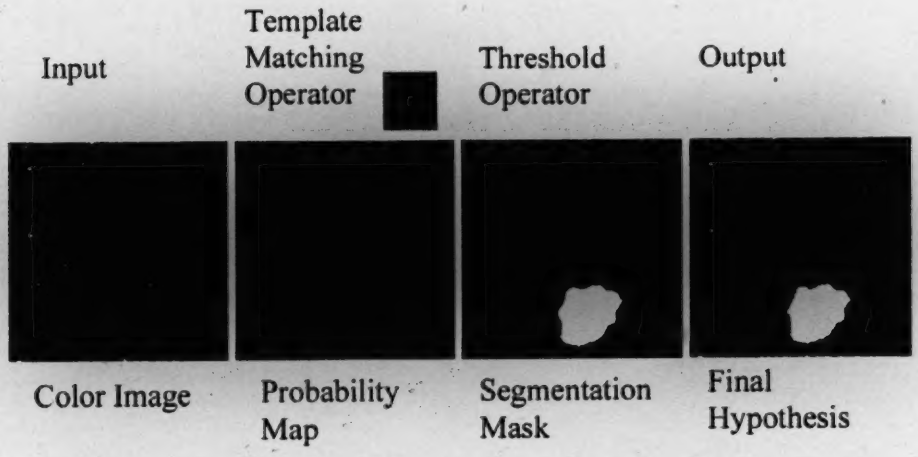


Figure A.23: The figure presents an example of a sequence of operators. First the input color image is used by a template matching operator to produce a probability map image indicating the degree of similarity between the template and the corresponding part of the input image. This particular probability map was obtained by performing a histogram matching operation, whereby the histogram of the template image is compared to a local histogram of a region of the image. The second operator in the sequence, uses the probability map to segment out the portion(s) of the image thought to contain the target object(s), in this case spruce trees. The final step simply presents the results in a more comprehensible fashion by overlaying the segmentation hypothesis over the input image.

Appendix B

Experimental Details and General Statistics of Interest

B.1 General Statistics

MR ADORE system uses the Intel IPL and OpenCv libraries consisting of about 200,000 lines of C/C++ code. The actual operator library used by MR ADORE needs an extra 40,000 lines of code that "wraps" the off-the-shelf IPL and OpenCv algorithms. In adition to using off-the-shelf libraries some of the algorithms were built from scratch (e.g. all the Template Matching routines). Finally, the actual MR ADORE system is composed of about 20,000 of C/C++ code and has an additional 5,000 lines of unix scripts facilitating efficient system activation and miscellaneous operations.

In order to train the system, we apply all possible operator sequences (constrained by a pre-specified maximal depth) to a number of training images. Figure B.1 shows general statistics of depth 4, 5, 6 search tree expansions (when pruning methods are used).

Table B.1: General Statistics expanding a single 256×256 image. **Sequence Length** - The *maximal* number of sequential operator applications allowed. **# of Nodes** - Total number of data tokens produced by applying all operator combinations of a given sequence length. **# of Sequences** - Total number of unique operator sequences of a given sequence length. **Directory Size** - Amount of storage needed to store all the data tokens produced by applying all unique operator sequences. **Time to Expand** - Amount of time needed to execute all unique operator sequences (of given length).

Sequence Length	# of Nodes	# of Sequences	Directory Size	Time to Expand
4	269	119	38Mb	30 sec
5	7382	3298	1Gb	10 min
6	192490	86037	26Gb	8 hrs

B.2 Off-line Scores for the plantation images

Table B.2 shows the scores/rewards obtained for the plantation images subject to a maximal sequence length constraints. Table B.3 shows the sequence index corresponding to optimal performance at a given search depth (i.e., maximal sequence length).

Table B.2: Off-line Optimal Scores for each of the 35 plantation images. **Note:** Images $12, 25, and 30$ do not contain target objects and therefore receive a perfect score of 1 when the system submits a blank/null hypothesis.

Image	Depth 4	Depth 5	Depth 6
1	0.294	0.362	0.455
2	0.536	0.539	0.57
3	0.349	0.518	0.556
4	0.345	0.396	0.558
5	0.356	0.447	0.482
6	0.463	0.564	0.603
7	0.371	0.509	0.539
8	0.509	0.574	0.579
9	0.553	0.569	0.64
10	0.591	0.653	0.654
11	0.478	0.621	0.645
12	1	1	1
13	0.218	0.441	0.559
14	0.416	0.536	0.587
15	0.439	0.515	0.559
16	0.373	0.535	0.592
17	0.335	0.472	0.57
18	0.326	0.484	0.589
19	0.422	0.508	0.53
20	0.586	0.64	0.671
21	0.411	0.564	0.565
22	0.352	0.485	0.53
23	0.351	0.509	0.55
24	0.278	0.349	0.445
25	1	1	1
26	0.517	0.618	0.709
27	0.3	0.328	0.447
28	0.338	0.526	0.589
29	0.347	0.407	0.435
30	1	1	1
31	0.188	0.417	0.486
32	0.309	0.502	0.542
33	0.349	0.507	0.604
34	0.317	0.438	0.46
35	0.27	0.388	0.433
Average Reward	**0.436771**	**0.5406**	**0.592371**

Table B.3: Off-line Optimal Sequences for each of the 35 plantation images (depth 4 and 5).

Image	Depth 4	Depth 5
1	13	6428
2	10	6967
3	65	1241
4	17	841
5	138	1250
6	16	1247
7	17	5735
8	13	1248
9	13	5735
10	12	1250
11	13	6006
12	159	7390
13	18	565
14	16	1253
15	248	6653
16	245	5735
17	18	1238
18	250	6698
19	18	1241
20	139	689
21	16	5736
22	250	6925
23	17	3401
24	17	6156
25	159	7390
26	138	2983
27	39	5648
28	18	1250
29	36	6992
30	159	7390
31	249	5946
32	12	6691
33	10	3668
34	250	6204
35	14	56
Number of Optimal Sequences	**18**	**28**

B.3 Hand-Crafted Features

The following feature sets were used in the experiments on plantation data.

RGB histogram A histogram of pixel intensity values was computed for each of the Red, Green, and Blue color channels [Gonzalez & E.Woods, 2002]. Each histogram, initially in the range of $[0 - 255]$ discrete intensity values, was re-quantized to 32 levels of brightness (resulting in a range of $[0 - 31]$. The 3 histograms were then concatenated to produce a feature vector of length 96. However as outlined in Chapter 4, feature vectors are composed of $f_{I+IM+I\tilde{M}}$, thus each vector is actually has $3(96) = 288$ elements.

RGB moments In a similar manner histograms of pixel intensity values were computed for each of the Red, Green, and Blue color channels [Gonzalez & E.Woods, 2002]. Each histogram, initially in the range of $[0 - 255]$ discrete intensity values, was re-quantized to 32 levels of brightness (resulting in a range of $[0 - 31]$. The first 3 moments [Gonzalez & E.Woods, 2002] were then extracted from each of the histograms. The 3 histograms were then concatenated to produce a feature vector of length 9. However as outlined in Chapter 4, feature vectors are composed of $f_{I+IM+I\tilde{M}}$, thus each vector is actually has $3(9) = 27$ elements.

HSV histograms Histograms were computed in exactly the same manner as the RGM histograms but on images converted to HSV (Hue, Saturation, Value) color space [Gonzalez & E.Woods, 2002].

HSV moments Analogous to RGB moments except produced from the HSV images.

Local Binary Patters Texture features were produced using the notion of local binary patterns [Ojala & Pietikinen, 1999]. The local binary pattern vector (produced from the grayscale image, using a3×3 lbp mask) has 256 elements yielding a feature vector of $3(256) = 768$ elements.

B.4 KNN distance metrics

L1 Manhattan Distance between vectors v_1 and v_2, defined by

$$\|v_1, v_2\|_{L1} = \sum_{i=0}^{n} |v_1(i) - v_2(i)| \tag{B.1}$$

L2 Euclidean Distance between vectors v_1 and v_2, defined by

$$\|v_1, v_2\|_{L2} = \sqrt{\sum_{i=0}^{n} (v_1(i) - v_2(i))^2} \tag{B.2}$$

Angle Angular Distance between vectors v_1 and v_2, defined by

$$\frac{v_1 \cdot v_2}{\|v_1\|_{L2}\|v_2\|_{L2}} \tag{B.3}$$

RL1 Relative Manhattan Distance between vectors v_1 and v_2, defined by

$$\|v_1, v_2\|_{L1} = \frac{\sum_{i=0}^{n} |v_1(i) - v_2(i)|}{\sum_{i=0}^{n} |v_1|} \tag{B.4}$$

RL2 Relative Euclidean Distance between vectors v_1 and v_2, defined by

$$\|v_1, v_2\|_{L2} = \frac{\sqrt{\sum_{i=0}^{n} (v_1(i) - v_2(i))^2}}{\sqrt{\sum_{i=0}^{n} (v_1(i))^2}} \tag{B.5}$$